Core System Package [Low Energy Controller volume]

Specification of the **Bluetooth**® System

Specification Volume 6

Covered Core Package Version: 5.0
Publication Date: Dec 06 2016

Revision History

The Revision History is shown in the [Vol 0] Part C, Appendix.

Contributors

The persons who contributed to this specification are listed in the [Vol 0] Part C, Appendix.

Web Site

This specification can also be found on the official Bluetooth web site: https://www.bluetooth.org/en-us/specification/adopted-specifications

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.[1]

Two modulation schemes are defined. The mandatory modulation scheme ("1 Msym/s modulation") uses a shaped, binary FM to minimize transceiver complexity. The symbol rate is 1 Msym/s. An optional modulation scheme ("2 Msym/s modulation") is similar but uses a symbol rate of 2 Msym/s.

The 1 Msym/s modulation supports two PHYs:

- LE 1M, with uncoded data at 1 Mb/s;
- LE Coded, with the Access Address, Coding Indicator, and TERM1 fields of the packet coded at 125 kb/s and the payload coded at either 125 kb/s or 500 kb/s.

A device shall support the LE 1M PHY. Support for the LE Coded PHY is optional.

The 2 Msym/s modulation supports a single PHY:

- LE 2M, with uncoded data at 2 Mb/s

A Time Division Duplex (TDD) scheme is used in both modes. This specification defines the requirements for a Bluetooth radio for the Low Energy radio.

Requirements are defined for two reasons:

- Provide compatibility between radios used in the system
- Define the quality of the system

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 A.1).

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.

1. Note that the transceiver defined in this Part does not meet the requirements for 'frequency hopping' in some governmental regulations. See the Bluetooth Low Energy Regulatory Aspects White Paper for more information.

The Bluetooth SIG maintains regulatory content associated with 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 300 440, EN 301 489-17

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 measurements, systems with an integral antenna should provide a temporary antenna connector during LE PHY 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)	100 mW (+20 dBm)

Table 3.1: Transmission power

Devices shall not exceed the maximum allowed transmit power levels set by the regulatory bodies that have jurisdiction over the locales in which the device is to be sold or intended to operate. Implementers should be aware that the maximum transmit power level permitted under a given set of regulations might not be the same for all modulation modes.

Note: The maximum output power for LE in v4.0, v4.1, and v4.2 is 10 mW.

Note: Using high transmit power in use cases where short ranges could be encountered may cause the receiver on the remote device to be saturated and result in link failure. Implementers should avoid use of high output power in such scenarios or employ a mechanism for switching between two or more transmit power levels in an attempt to establish, re-establish, or maintain connections.

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

Bluetooth devices may be informatively classified into power classes based on the highest output power the LE PHY supports, as defined in Table 3.2.

Power Class	**Maximum Output Power (P_{max})**	**Minimum Output Power**[1]
1	100 mW (+20 dBm)	10 mW (+10 dBm)
1.5	10 mW (+10 dBm)	0.01 mW (-20 dBm)
2	2.5 mW (+4 dBm)	0.01 mW (-20 dBm)
3	1 mW (0 dBm)	0.01 mW (-20 dBm)

Table 3.2: LE PHY power classes

1. Minimum output power at maximum power setting

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.

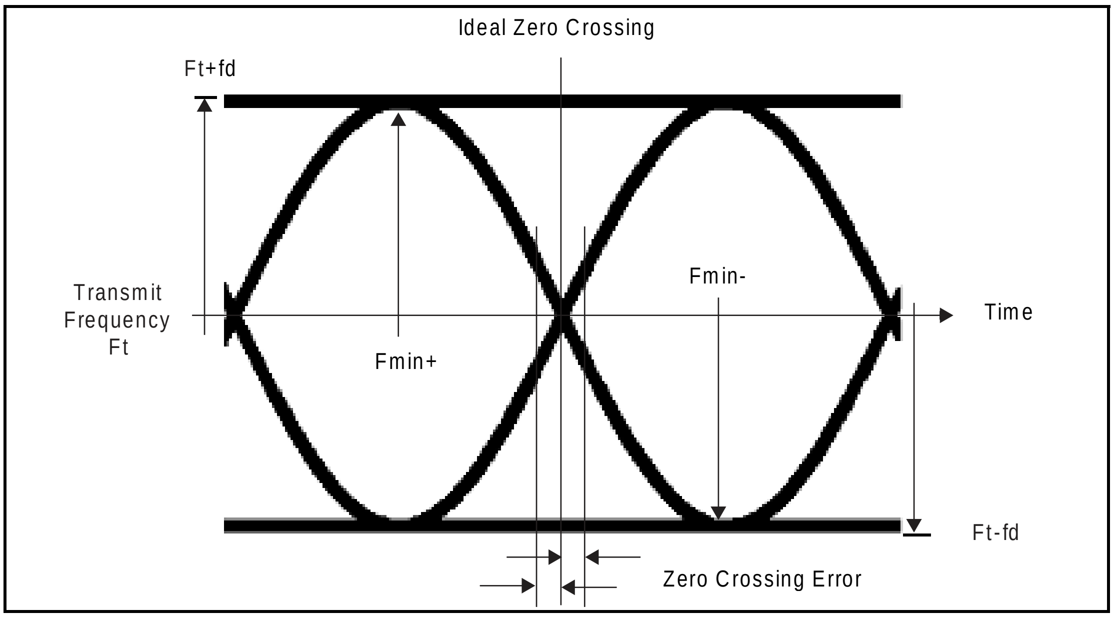


Figure 3.1: GFSK parameters definition

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.

The minimum frequency deviation shall never be less than 185 kHz when transmitting at 1 megasymbol per second (Msym/s) symbol rate and never be less than 370 kHz when transmitting at 2 Msym/s symbol rate. 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.

3.1.1 Stable Modulation Index

An LE device with a transmitter that has a stable modulation index may inform the receiving LE device of this fact through the feature support mechanism (see [Vol 6] Part B, Section 4.6). The modulation index for these transmitters shall be between 0.495 and 0.505. A device shall only state that it has a stable modulation index if that applies to all LE transmitter PHYs it supports.

A transmitter that does not have a stable modulation index is said to have a standard modulation index.

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 when transmitting with1 Msym/s modulation (applies to the LE 1M and LE Coded PHYs) or at least 4 MHz from the carrier when transmitting with 2 Msym/s modulation (applies to the LE 2M PHY). 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 3.3: Transmit Spectrum mask when transmitting with 1 Msym/s modulation

Frequency offset	**Spurious Power**
4 MHz (|M-N| = 4)	-20 dBm
5 MHz (|M-N| = 5)	-20 dBm
6 MHz or greater (|M-N| ≥ 6)	-30 dBm

Table 3.4: Transmit Spectrum mask when transmitting with 2 Msym/s modulation

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 3.5.

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

Table 3.5: 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 the BER specified in Table 4.1 is achieved.

Maximum Supported Payload Length (bytes)	BER (%)
≤ 37	0.1
≥ 38 and ≤ 63	0.064
≥ 64 and ≤ 127	0.034
≥ 128	0.017

Table 4.1: Actual sensitivity BER by maximum payload length

The actual sensitivity level of the receiver for a given PHY shall be as specified in Table 4.2. This shall apply 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

PHY	Sensitivity (dBm)
LE Uncoded PHYs	≤ -70
LE Coded PHY with S=2 coding	≤ -75
LE Coded PHY with S=8 coding	≤ -82

Table 4.2: Receiver sensitivity for a given PHY

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. 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 4.3, Table 4.4, Table 4.5, and Table 4.6:

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 4.3: Interference performance for LE 1M PHY

Frequency of Interference	**Ratio**
Co-Channel interference, $C/I_{co\text{-}channel}$	21 dB
Adjacent (2 MHz) interference[1], $C/I_{2\ MHz}$	15 dB
Adjacent (4 MHz) interference[1], $C/I_{4\ MHz}$	-17 dB
Adjacent (≥6 MHz) interference[1], $C/I_{\geq 6\ MHz}$	-27 dB
Image frequency interference[1 2 4], C/I_{Image}	-9 dB
Adjacent (2 MHz) interference to in-band image frequency[1], $C/I_{Image\pm 2MHz}$	-15 dB

Table 4.4: Interference performance for LE 2M PHY

Frequency of Interference	**Ratio**
Co-Channel interference, $C/I_{co\text{-}channel}$	12 dB
Adjacent (1 MHz) interference[1], $C/I_{1\ MHz}$	6 dB
Adjacent (2 MHz) interference[1], $C/I_{2\ MHz}$	-26 dB

Table 4.5: Interference performance for the LE Coded PHY with S=8 coding (125 kb/s data rate)

Frequency of Interference	**Ratio**
Adjacent (≥3 MHz) interference[1], $C/I_{\geq 3\ MHz}$	-36 dB
Image frequency interference[1 2 3], C/I_{Image}	-18 dB
Adjacent (1 MHz) interference to in-band image frequency[1], $C/I_{Image\pm 2MHz}$	-24 dB

Table 4.5: Interference performance for the LE Coded PHY with S=8 coding (125 kb/s data rate)

Frequency of Interference	**Ratio**
Co-Channel interference, $C/I_{co\text{-}channel}$	17 dB
Adjacent (1 MHz) interference[1], $C/I_{1\ MHz}$	11 dB
Adjacent (2 MHz) interference[1], $C/I_{2\ MHz}$	-21 dB
Adjacent (≥3 MHz) interference[1], $C/I_{\geq 3\ MHz}$	-31 dB
Image frequency interference[1 2 3], C/I_{Image}	-13 dB
Adjacent (1 MHz) interference to in-band image frequency[1], $C/I_{Image\pm 2MHz}$	-19 dB

Table 4.6: Interference performance for the LE Coded PHY with S=2 coding (500 kb/s data rate)

Notes:

1. If two adjacent frequency specifications from Table 4.3, Table 4.4, Table 4.5, or Table 4.6 (as appropriate) 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.
4. If the image frequency ≠ n*2 MHz, then the image reference frequency is defined as the closest n*2 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 when receiving with 1 Msym/s modulation and a distance of ≥4 MHz when receiving with 2 Msym/s modulation; different spurious response channels are allowed for the two modulation schemes. This excludes the image frequency with both 1 Msym/s and 2 Msym/s modulation, the image frequency ±1MHz with 1 Msym/s modulation, and the image frequency ±2 MHz with 2 Msym/s modulation. On these spurious response RF channels, a relaxed interference requirement C/I = -17 dB shall be met by both 1 Msym/s and 2 Msym/s modulation transmitters.

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 2426 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 4.7: 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.

When receiving with 1 Msym/s modulation, frequencies f_0, f_1 and f_2 shall be chosen such that $f_0 = 2*f_1 - f_2$ and
$| f_2-f_1 | = n * 1$ MHz, where n can be 3, 4, or 5.

When receiving with 2 Msym/s modulation, frequencies f_0, f_1 and f_2 shall be chosen such that
$f_0 = 2*f_1 - f_2$ and
$| f_2-f_1 | = n * 2$ MHz, where n can be 3, 4, or 5.

The system shall fulfill at least one of the three alternatives (n=3, 4, or 5); different modulation schemes can use different alternatives.

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

The reference signal for LE is defined as:

Modulation = GFSK

Modulation index = 0.5 ± 1% for standard modulation index, 0.5 ± 0.5% for stable modulation index

BT = 0.5 ± 1%

Data Bit Rate =

- 1 Mb/s ±1 ppm for the LE 1M PHY
- 2 Mb/s ±1 ppm for the LE 2M PHY
- 125 kb/s ±1 ppm for the LE Coded PHY when using S=8 coding
- 500 kb/s ±1 ppm for the LE Coded PHY when using S=2 coding

Modulating Data for wanted signal = PRBS9

Modulating Data for interfering signal = PRBS15

Frequency accuracy better than ±1 ppm

4.7 STABLE MODULATION INDEX

An LE device may have a receiver that can take advantage of the fact that the remote device indicates support for the Stable Modulation Index - Transmitter feature (see [Vol 6] Part B, Section 4.6). Such a receiver is said to have stable modulation index support.

APPENDIX A TEST CONDITIONS

A.1 NORMAL OPERATING CONDITIONS (NOC)

A.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.

A.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.

Core System Package [Low Energy Controller volume]

Part B

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.

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.

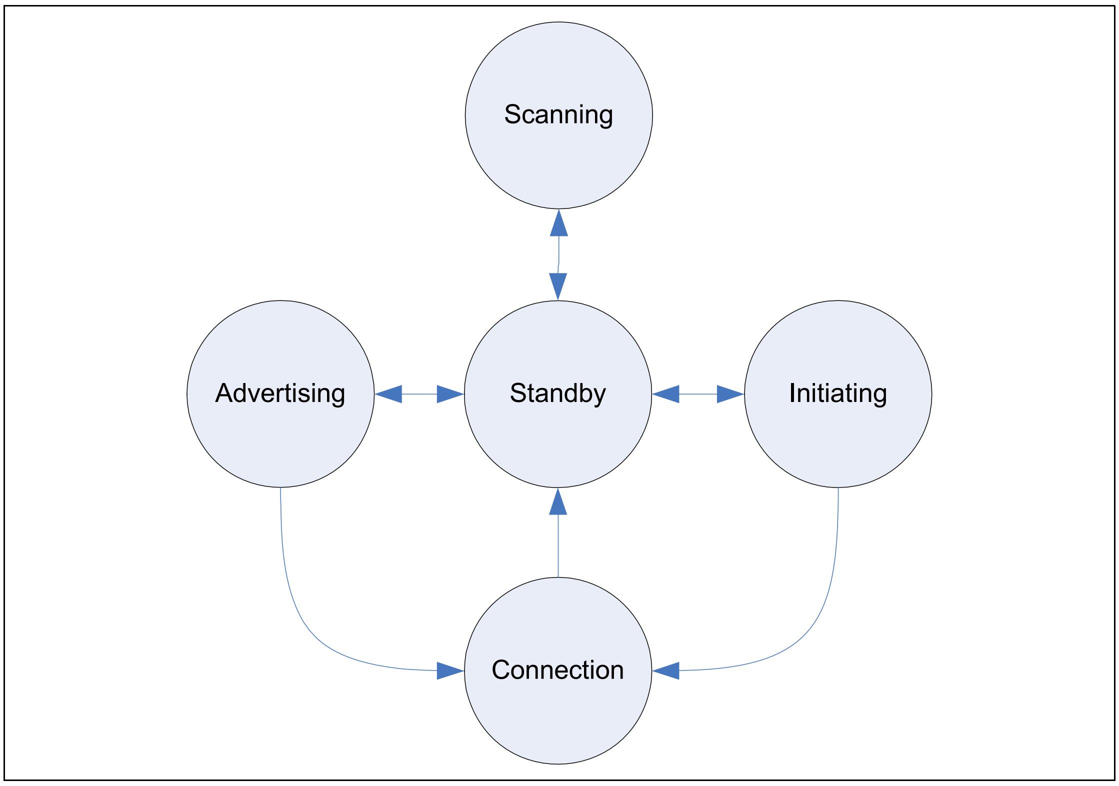


Figure 1.1: State diagram of the Link Layer state machine

1.1.1 Permitted State and Role Combination Restrictions

The Link Layer may optionally support multiple state machines. If it does support multiple state machines, then:

- The Link Layer in the Connection State may operate in the Master Role and Slave Role at the same time.
- The Link Layer in the Connection State operating in the Slave Role may have multiple connections.
- The Link Layer in the Connection State operating in the Master Role may have multiple connections.
- All other combinations of states and roles may also be supported.
- The Link Layer in the Connection State shall have at most one connection to another Link Layer in the Connection State.

A Link Layer implementation is not required to support all the possible state combinations that are allowed by this specification. However, if it supports a state or state combination given in the "combination A" column of Table 1.1, it shall also support the corresponding state or state combination in the "combination B" column.

Combination A	Combination B
Initiating plus any combination C of other states	Connection (Master role) plus the same combination C
Connection (Master role) plus Initiating plus any combination C of other states	Connection (Master role) to more than one device in the slave role plus the same combination C
Connectable or a directed advertising state plus any combination C of other states	Connection (Slave role) plus the same combination C
Connection (Slave role) plus Connectable or a directed advertising state plus any combination C of other states	Connection (Slave role) to more than one device in the Master role plus the same combination C

Table 1.1: Requirements on supported states and state combinations

In each case, the combination of other states C may be empty. Note that, in the last two rows, "other states" includes other Connectable or directed advertising states.

1.1.2 Devices supporting only some states

Devices supporting only some link layer states or only one of the two roles within the Connection State are not required to support features (including supporting particular PDUs, procedures, data lengths, or HCI commands or particular features of an HCI command) that are only used by a state or mode that the device does not support.

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:

$b_0b_1b_2 = 110$

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., the advertising channel Access Address in Section 2.1.2) are written with the MSB to the left. There are two basic formats: one for the LE Uncoded PHYs, described in Section 2.1, and one for the LE Coded PHY, described in Section 2.2.

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

A device's Identity Address is a Public Device Address or Random Static Device Address that it uses in packets it transmits. If a device is using Resolvable Private Addresses, it shall also have an Identity Address.

1.3.1 Public Device Address

The public device address shall be created in accordance with [Vol 2] Part B, Section 1.2, with the exception that the restriction on LAP values does not apply unless the public device address will also be used as a BD_ADDR for a BR/EDR Controller.

1.3.2 Random Device Address

The random device address may be of either of the following two sub-types:

- Static address
- Private address.

The term random device address refers to both static and private address types.

The transmission of a random device address is optional. A device shall accept the reception of a random device address from a remote device.

1.3.2.1 Static Device Address

A static address is a 48-bit randomly generated address and shall meet the following requirements:

- The two most significant bits of the address shall be equal to 1
- At least one bit of the random part of the address shall be 0
- At least one bit of the random part of the address shall be 1

The format of a static address is shown in Figure 1.2.

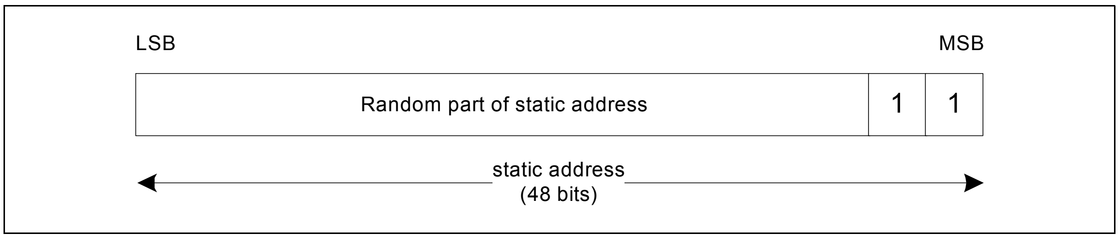


Figure 1.2: Format of static address

A device may choose to initialize its static address to a new value after each power cycle. A device shall not change its static address value once initialized until the device is power cycled.

Note: If the static address of a device is changed, then the address stored in peer devices will not be valid and the ability to reconnect using the old address will be lost.

1.3.2.2 Private Device Address Generation

The private address may be of either of the following two sub-types:

- Non-resolvable private address
- Resolvable private address

To generate a non-resolvable address, the device shall generate a 48-bit address with the following requirements:

- The two most significant bits of the address shall be equal to 0
- At least one bit of the random part of the address shall be 1
- At least one bit of the random part of the address shall be 0
- The address shall not be equal to the public address

The format of a non-resolvable private address is shown in Figure 1.3.

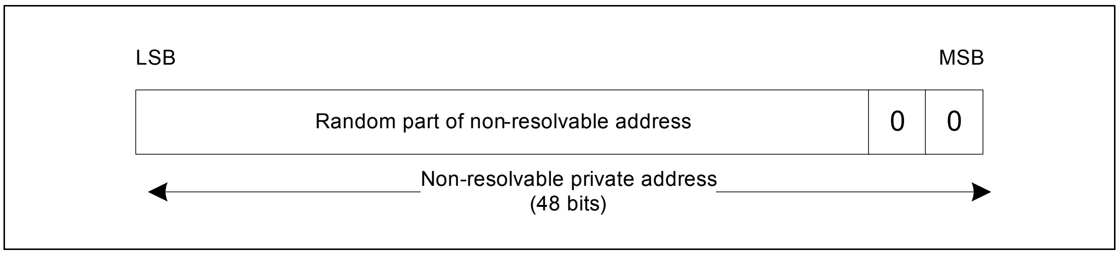

LSB
MSB
Random part of non-resolvable address
0
0
Non-resolvable private address
(48 bits)

Figure 1.3: Format of non-resolvable private address

To generate a resolvable private address, the device must have either the Local Identity Resolving Key (IRK) or the Peer Identity Resolving Key (IRK). The resolvable private address shall be generated with the IRK and a randomly generated 24-bit number. The random number is known as *prand* and shall meet the following requirements:

- The two most significant bits of *prand* shall be equal to 0 and 1 as shown in Figure 1.4
- At least one bit of the random part of *prand* shall be 0
- At least one bit of the random part of *prand* shall be 1

The format of the resolvable private address is shown in Figure 1.4.

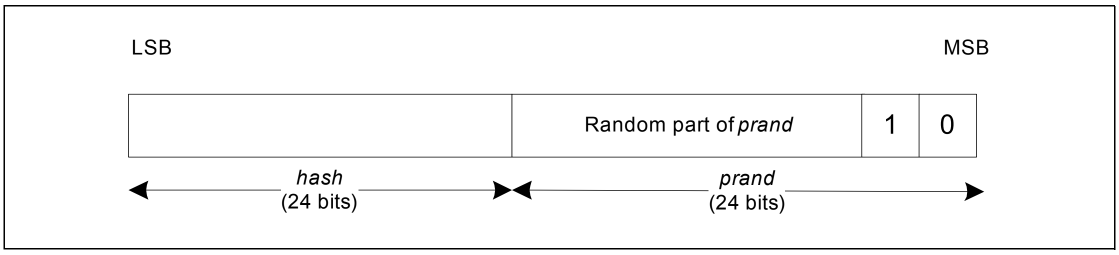


Figure 1.4: Format of resolvable private address

The hash is generated using the random address function *ah* defined in [Vol 3] Part H, Section 2.2.2 with the input parameter *k* set to the device's IRK and the input parameter *r* set to *prand*.

$$hash = ah(\mathrm{IRK}, prand)$$

The prand and hash are concatenated to generate the random address (randomAddress) in the following manner:

$$randomAddress = hash \,||\, prand$$

The least significant octet of *hash* becomes the least significant octet of *randomAddress* and the most significant octet of *prand* becomes the most significant octet of *randomAddress*.

1.3.2.3 Private Device Address Resolution

A resolvable private address may be resolved if the corresponding device's IRK is available using this procedure. If a resolvable private address is resolved, the device can associate this address with the peer device.

The resolvable private address (*RPA*) is divided into a 24-bit random part (*prand*) and a 24-bit hash part (*hash*). The least significant octet of the *RPA* becomes the least significant octet of *hash* and the most significant octet of *RPA* becomes the most significant octet of *prand*. A *localHash* value is then generated using the random address hash function *ah* defined in [Vol 3] Part H, Section 2.2.2 with the input parameter *k* set to IRK of the known device and the input parameter *r* set to the *prand* value extracted from the *RPA*.

$$localHash = ah(\mathrm{IRK}, prand)$$

The *localHash* value is then compared with the *hash* value extracted from *RPA*. If the *localHash* value matches the extracted *hash* value, then the identity of the peer device has been resolved.

If a device has more than one stored IRK, the device repeats the above procedure for each stored IRK to determine if the received resolvable private address is associated with a stored IRK, until either address resolution is successful for one of the IRKs or all have been tried.

Note: A device that cannot resolve a private address within T_IFS may respond on the reception of the next event.

A non-resolvable private address cannot be resolved.

1.4 PHYSICAL CHANNEL

As specified in [Vol 6] Part A, Section 2, 40 RF channels are defined in the 2.4GHz ISM band. These RF channels are allocated into three LE physical channels: advertising, periodic, and data. The advertising physical channel uses all 40 RF channels for discovering devices, initiating a connection and broadcasting data. These RF channels are divided into 3 RF channels, known as the "primary advertising channel", used for initial advertising and all legacy advertising activities, and 37 RF channels, known as the "secondary advertising channel", used for the majority of the communications involved. 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). The periodic physical channel uses the same RF channels as the secondary advertising channel over the advertising physical channel.

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' on the data physical channel or ‘synchronized’ to the periodic physical channel (whether or not it is actively involved in communications over the channel).

1.4.1 Advertising and Data Channel Indices

Table 1.2 shows the mapping from PHY Channel to Channel Index and Channel Type. An '●' in the table below indicates the PHY channel and index are used by the specified channel type.

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

Table 1.2: Mapping of PHY Channel to Channel Index and Channel Type

2 AIR INTERFACE PACKETS

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

2.1 PACKET FORMAT FOR THE LE UNCODED PHYS

The following packet format is defined for the LE Uncoded PHYs (LE 1M and LE 2M) and is used for both advertising channel packets and data channel packets.

This packet format is shown in Figure 2.1. Each packet consists of four mandatory fields. The mandatory fields are Preamble, Access Address, PDU, and CRC.

LSB MSB

Preamble	Access Address	PDU	CRC
(1 or 2 octets)	(4 octets)	(2 to 257 octets)	(3 octets)

Figure 2.1: Link Layer packet format for the LE Uncoded PHYs

The preamble is 1 octet when transmitting or receiving on the LE 1M PHY and 2 octets when transmitting or receiving on the LE 2M PHY. The Access Address is 4 octets. The PDU range is from 2 to 257 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 entire packet is transmitted at the same symbol rate (either 1 Msym/s or 2 Msym/s modulation).

Packets take between 44 and 2120 μs to transmit.

2.1.1 Preamble

All Link Layer packets have a preamble which is used in the receiver to perform frequency synchronization, symbol timing estimation, and Automatic Gain Control (AGC) training. The preamble is a fixed sequence of alternating 0 and 1 bits. For packets transmitted on the LE 1M PHY, the preamble is 8 bits; for packets transmitted on the LE 2M PHY, the preamble is 16 bits. The first bit of the preamble (in transmission order) shall be the same as the LSB of the Access Address. The preamble is shown in Figure 2.2.

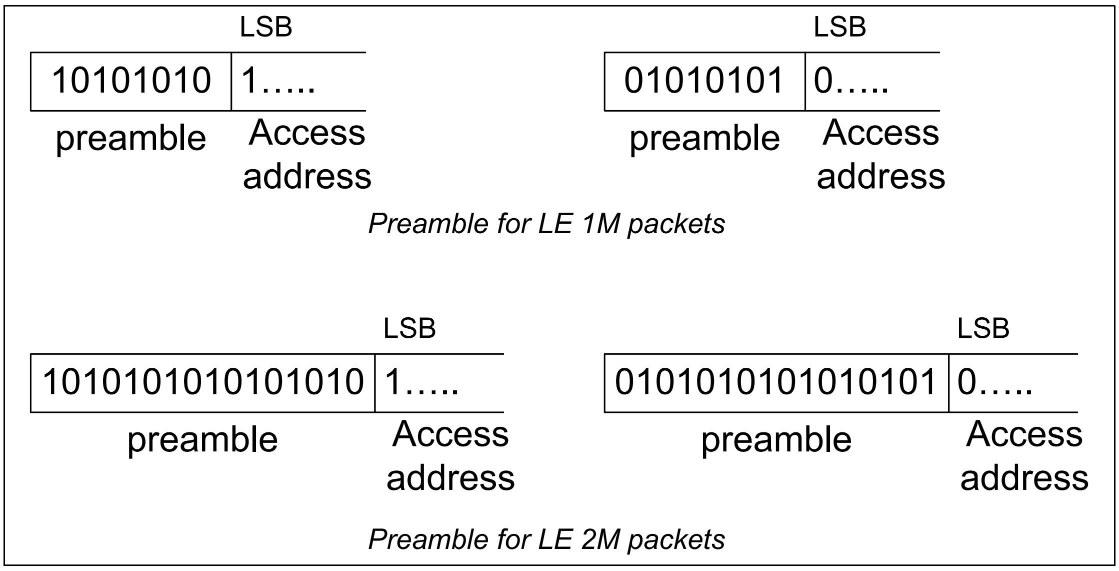


Figure 2.2: Preamble

2.1.2 Access Address

The AUX_SYNC_IND PDU, and any AUX_CHAIN_IND PDUs connected to it, shall use the Access Address (AA) value set in the SyncInfo field (see Section 2.3.4.6) contained in the AUX_ADV_IND PDU that describes the periodic advertising.

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

It is intended that each Link Layer connection between any two devices and each periodic advertisement has a different Access Address.

The Link Layer in the Initiating State shall generate a new Access Address for each initiating PDU it sends (see Section 2.3.3.1). The Link Layer in the Advertising State shall generate a new Access Address each time that it enables periodic advertising on an advertising set. The address is sent in the SyncInfo field (see Section 2.3.4.6) of PDUs referring to that periodic advertising.

The Access Address shall be a 32-bit value. Each time it needs a new Access Address, the Link Layer shall generate a new random value that meets the following requirements:

- It shall not be the Access Address for any existing Link Layer connection on this device.
- It shall not be the Access Address for any enabled periodic advertising.
- 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.

The seed for the random number generator shall be from a physical source of entropy and should have at least 20 bits of entropy.

If the random number does not meet the above requirements, new random numbers shall be generated until the requirements are met.

On an implementation that also supports the LE Coded PHY (see Section 2.2, the Access Address shall also meet the following requirements:

- It shall have at least three ones in the least significant 8 bits.
- It shall have no more than eleven transitions in the least significant 16 bits.

2.1.3 PDU

The preamble and Access Address are followed by a PDU. When a packet is transmitted on either the primary or secondary advertising channel, the PDU shall be the Advertising Channel PDU as defined in Section 2.3. When a packet is transmitted on the 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 PACKET FORMAT FOR THE LE CODED PHY

The following packet format is defined for the LE Coded PHY and is used for both advertising channel packets and data channel packets. This packet format is shown in Figure 2.3.

.

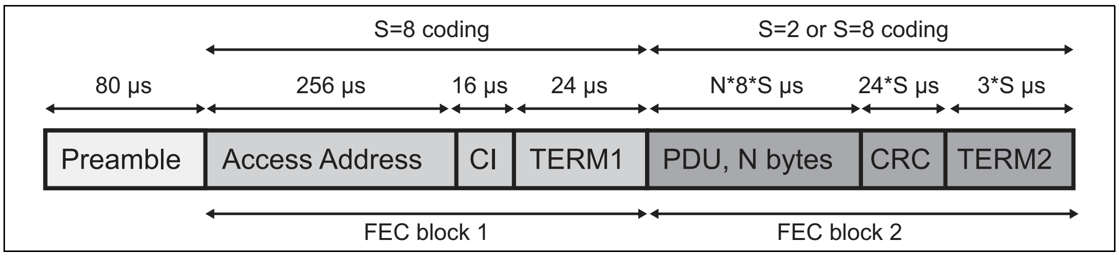


Figure 2.3: Link Layer packet format for the LE Coded PHY

Each packet consists of the Preamble, FEC block 1, and FEC block 2.

The Preamble is not coded.

The FEC block 1 consists of three fields: Access Address, Coding Indicator (CI), and TERM1. These shall use the S=8 coding scheme as defined in Section 3.3.1.

The CI field determines which coding scheme is used for FEC block 2.

The FEC block 2 consists of three fields: PDU, CRC, and TERM2. These shall use either the S=2 or S=8 coding scheme as defined in Section 3.3, depending on the value of the CI field.

The entire packet is transmitted with 1 Msym/s modulation.

Table 2.1 captures the size and duration of the data packet fields.

	Fields						
	Preamble	**Access Address**	**CI**	**TERM1**	**PDU**	**CRC**	**TERM2**
Number of Bits	Uncoded	32	2	3	16 – 2056	24	3
Duration when using S=8 coding (µs)	80	256	16	24	128 – 16448	192	24
Duration when using S=2 coding (µs)	80	256	16	24	32 – 4112	48	6

Table 2.1: LE Coded PHY field sizes and durations in microseconds

Packets take between 462 and 17040 µs to transmit.

2.2.1 Preamble

The Preamble is 80 symbols in length and consists of 10 repetitions of the symbol pattern '00111100' (in transmission order).

2.2.2 Access Address

The Access Address is specified in Section 2.1.2.

2.2.3 Coding Indication

The CI field consists of two bits as defined in Table 2.2.

CI Field	Meaning
00b	FEC Block 2 coded using S=8
01b	FEC Block 2 coded using S=2
All other values	Reserved for future use

Table 2.2: Meaning of CI field

2.2.4 PDU

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

2.2.5 CRC

The CRC is 24 bits in length and the value is calculated over all the PDU bits. The CRC generator polynomial is defined in Section 3.1.1.

2.2.6 TERM1 and TERM2

There is a termination field at the end of each FEC block referred to as TERM1 and TERM2. Each termination field is 3 bits long and forms the termination sequence defined in Section 3.3.1.

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 2.4. The 16-bit Header field of the advertising channel PDU is as shown in Figure 2.5.

LSB	MSB
Header (16 bits)	Payload (1-255 octets)

Figure 2.4: Advertising channel PDU

LSB					MSB
PDU Type (4 bits)	RFU (1 bit)	ChSel (1 bit)	TxAdd (1 bit)	RxAdd (1 bit)	Length (8 bits)

Figure 2.5: 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 2.3. This table also shows which channel and which PHYs the packet may appear on.

			Permitted PHYs		
PDU Type	**PDU Name**	**Channel**	**LE 1M**	**LE 2M**	**LE Coded**
0000b	ADV_IND	Primary Advertising	●		
0001b	ADV_DIRECT_IND	Primary Advertising	●		
0010b	ADV_NONCONN_IND	Primary Advertising	●		
0011b	SCAN_REQ	Primary Advertising	●		
	AUX_SCAN_REQ	Secondary Advertising	●	●	●
0100b	SCAN_RSP	Primary Advertising	●		
0101b	CONNECT_IND	Primary Advertising	●		
	AUX_CONNECT_REQ	Secondary Advertising	●	●	●
0110b	ADV_SCAN_IND	Primary Advertising	●		

Table 2.3: Advertising channel PDU Header's PDU Type field encoding

PDU Type	PDU Name	Channel	Permitted PHYs		
			LE 1M	LE 2M	LE Coded
0111b	ADV_EXT_IND	Primary Advertising	●		●
	AUX_ADV_IND	Secondary Advertising	●	●	●
	AUX_SCAN_RSP	Secondary Advertising	●	●	●
	AUX_SYNC_IND	Secondary Advertising	●	●	●
	AUX_CHAIN_IND	Secondary Advertising	●	●	●
1000b	AUX_CONNECT_RSP	Secondary Advertising	●	●	●
All other values	Reserved for Future Use				

Table 2.3: Advertising channel PDU Header's PDU Type field encoding

Each PDU shall only appear on those PHYs indicated with a bullet '●' and on the Advertising Channel shown in Table 2.3.

The ChSel, 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 ChSel, 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 1 to 255 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 2.3.4. The PDU Types marked as Reserved for future use shall not be sent and shall be ignored upon receipt.

Within advertising channel PDUs, advertising data or scan response 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 11.

Some advertising channel PDUs contain an AuxPtr field (see Section 2.3.4.5) which points to a packet containing another advertising channel PDU. In this case, the second packet and PDU are the *auxiliary packet* and *auxiliary PDU* of the original PDU, which in turn is the *superior packet* and *superior PDU* of the second one. Note that a PDU can only have one auxiliary PDU but more than one superior PDU.

Given a packet, its *subordinate set* consists of its auxiliary packet, if any, and the subordinate set of the auxiliary packet. A packet without an AuxPtr field has an empty subordinate set.

2.3.1 Advertising PDUs

The following advertising channel PDU Types are called advertising PDUs:

- ADV_IND
- ADV_DIRECT_IND
- ADV_NONCONN_IND
- ADV_SCAN_IND
- ADV_EXT_IND
- AUX_ADV_IND
- AUX_SYNC_IND
- AUX_CHAIN_IND

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. The ADV_IND, ADV_DIRECT_IND, ADV_NONCONN_IND, and ADV_SCAN_IND PDUs are called “legacy advertising PDUs”. The ADV_EXT_IND, AUX_ADV_IND, AUX_SYNC_IND, and AUX_CHAIN_IND PDUs are called “extended advertising PDUs”. Advertising events using legacy advertising PDUs are called “legacy advertising events”.

2.3.1.1 ADV_IND

The ADV_IND PDU has the Payload as shown in Figure 2.6. The PDU shall be used in connectable and scannable undirected advertising events. The TxAdd in the advertising channel PDU header indicates whether the advertiser’s address in the AdvA field is public (TxAdd = 0) or random (TxAdd = 1). The ChSel field in the advertising channel PDU header shall be set to 1 if the advertiser supports the LE Channel Selection Algorithm #2 feature (see Section 4.5.8.3).

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

Figure 2.6: 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 2.7. The PDU shall be used in connectable directed advertising events. The TxAdd in the advertising channel PDU header indicates whether the advertiser's address in the AdvA field is public (TxAdd = 0) or random (TxAdd = 1). The RxAdd in the advertising channel PDU header indicates whether the target's address in the TargetA field is public (RxAdd = 0) or random (RxAdd = 1). The ChSel field in the advertising channel PDU header shall be set to 1 if the advertiser supports the LE Channel Selection Algorithm #2 feature (see Section 4.5.8.3).

Payload	
AdvA (6 octets)	TargetA (6 octets)

Figure 2.7: ADV_DIRECT_IND PDU Payload

The Payload field consists of AdvA and TargetA fields. The AdvA field shall contain the advertiser's public or random device address as indicated by TxAdd. The TargetA field is the address of the device to which this PDU is addressed. The TargetA field shall contain the target'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 2.8. The PDU shall be used in non-connectable and non-scannable undirected advertising events. The TxAdd in the advertising channel PDU header 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 2.8: 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

The ADV_SCAN_IND PDU has the Payload as shown in Figure 2.9. The PDU shall be used in scannable undirected advertising events. The TxAdd in the advertising channel PDU header 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 2.9: 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.

2.3.1.5 ADV_EXT_IND

The ADV_EXT_IND PDU uses the Common Extended Advertising Payload Format described in Section 2.3.4. The PDU may be used in all advertising events (except connectable and scannable undirected) as indicated by the AdvMode field value. An advertising event using an ADV_EXT_IND PDU is directed if, and only if, either the TargetA field is present or the AuxPtr field is present and points to a PDU where the TargetA field is present.

The Common Extended Advertising Payload Format fields permitted in the ADV_EXT_IND PDU are shown in Table 2.4.

		Common Extended Advertising Payload Format fields							
Event Type	**Adv Mode**	**AdvA**	**TargetA**	**ADI**	**Aux Ptr**	**Sync Info**	**Tx Power**	**ACAD**	**Adv Data**
Non-Connectable and Non-Scannable Undirected without auxiliary packet	00b	M	X	X	X	X	O	X	X
Non-Connectable and Non-Scannable Undirected with auxiliary packet	00b	C1	X	M	M	X	C1	X	X

Table 2.4: Common Extended Advertising Payload Format fields permitted in the ADV_EXT_IND PDU

		Common Extended Advertising Payload Format fields							
Event Type	**Adv Mode**	**AdvA**	**TargetA**	**ADI**	**Aux Ptr**	**Sync Info**	**Tx Power**	**ACAD**	**Adv Data**
Non-Connectable and Non-Scannable Directed without auxiliary packet	00b	M	M	X	X	X	O	X	X
Non-Connectable and Non-Scannable Directed with auxiliary packet	00b	C1	C1	M	M	X	C1	X	X
Connectable Undirected	01b	X	X	M	M	X	C1	X	X
Connectable Directed	01b	X	X	M	M	X	C1	X	X
Scannable Undirected	10b	X	X	M	M	X	C1	X	X
Scannable Directed	10b	X	X	M	M	X	C1	X	X
RFU	11b								

Table 2.4: Common Extended Advertising Payload Format fields permitted in the ADV_EXT_IND PDU

For the non-connectable and non-scannable directed and non-connectable and non-scannable undirected event types without ACAD or AdvData, the Controller can choose whether or not to use an auxiliary packet. See Sections 4.4.2.6 and 4.4.2.9.

In all tables (including subsequent sections) describing permitted Common Extended Advertising Payload Format Fields:

M	This field is mandatory.
O	This field is optional.
X	This field is reserved for future use.
C1	This field is optional on the LE 1M PHY and reserved for future use on the LE Coded PHY.
C2	This field is mandatory if the corresponding field in the ADV_EXT_IND PDU is not present, otherwise it is reserved for future use.
C3	This field is mandatory if the corresponding field in the PDU pointing to this PDU is present, otherwise it is reserved for future use.
C4	This field is optional if the corresponding field in the ADV_EXT_IND PDU is not present, otherwise it is reserved for future use.

Fields reserved for future use shall not be present when the packet is sent and shall be ignored when received.

Any auxiliary packet shall be an AUX_ADV_IND packet with the same AdvMode as the ADV_EXT_IND packet.

2.3.1.6 AUX_ADV_IND

The AUX_ADV_IND PDU uses the Common Extended Advertising Payload Format described in Section 2.3.4. The PDU may be used in all advertising events (except connectable and scannable undirected) as indicated by the AdvMode field value.

The AdvMode field indicates the type of advertising event the AUX_ADV_IND packet is being used for.

The Common Extended Advertising Payload Format fields permitted in the AUX_ADV_IND PDU are shown in Table 2.5.

The PHY used for the AUX_ADV_IND shall be specified in the AuxPtr field of the superior PDU. The PHY specified in any AuxPtr field in an AUX_ADV_IND PDU shall be the same as the PHY the PDU was sent on.

The ADI field shall have the same value as the field in the PDU pointing to this PDU. Note: The ADI field can be used to detect collisions.

Any auxiliary PDU shall be an AUX_CHAIN_IND PDU.

The SyncInfo field, when present, shall point to an AUX_SYNC_IND PDU.

		Common Extended Advertising Payload Format fields							
Adv Mode	**Event Type**	**AdvA**	**TargetA**	**ADI**	**Aux Ptr**	**Sync Info**	**Tx Power**	**ACAD**	**Adv Data**
00b	Non-Connectable and Non-Scanna-ble Undirected	C4	X	M	O	O	O	O	O
00b	Non-Connectable and Non-Scanna-ble Directed	C4	C2	M	O	O	O	O	O
01b	Connectable Undirected	M	X	M	X	X	O	O	O
01b	Connectable Directed	M	M	M	X	X	O	O	O
10b	Scannable Undirected	M	X	M	X	X	O	O	X
10b	Scannable Directed	M	M	M	X	X	O	O	X
11b	RFU								

Table 2.5: Common Extended Advertising Payload Format fields permitted in the AUX_ADV_IND PDU

2.3.1.7 AUX_SYNC_IND

The AUX_SYNC_IND PDU uses the Common Extended Advertising Payload Format described in Section 2.3.4. The PDU is used in periodic advertising.

The AdvMode field shall be set to 00b.

The Common Extended Advertising Payload Format fields permitted in the AUX_SYNC_IND PDU are shown in Table 2.6.

The PHY used for the AUX_SYNC_IND PDU shall be that specified in Section 4.4.2.12.

Any auxiliary PDU shall be an AUX_CHAIN_IND PDU.

		Common Extended Advertising Payload Format fields							
Adv Mode	**Event Type**	**AdvA**	**TargetA**	**ADI**	**Aux Ptr**	**Sync Info**	**Tx Power**	**ACAD**	**Adv Data**
00b	Non-Connectable and Non-Scanna-ble Undirected or Directed	X	X	X	O	X	O	O	O
01b – 11b	RFU								

Table 2.6: Common Extended Advertising Payload Format fields permitted in the AUX_SYNC_IND PDU

2.3.1.8 AUX_CHAIN_IND

The AUX_CHAIN_IND PDU uses the Common Extended Advertising Payload Format described in Section 2.3.4. The PDU is used to hold additional AdvData. Its superior PDU is an AUX_ADV_IND, AUX_SYNC_IND, AUX_SCAN_RSP or another AUX_CHAIN_IND PDU.

The AdvMode field shall be set to 00b.

The Common Extended Advertising Payload Format fields permitted in the AUX_CHAIN_IND PDU are shown in Table 2.7.

The PHY used for the AUX_CHAIN_IND PDU shall be the same as the PHY used for its superior PDU.

The ADI field, when present, shall have the same value as the field in the superior PDU. Note: The ADI field can be used to detect collisions.

Any auxiliary PDU shall be another AUX_CHAIN_IND PDU.

		Common Extended Advertising Payload Format fields							
Adv Mode	**Event Type**	**AdvA**	**TargetA**	**ADI**	**Aux Ptr**	**Sync Info**	**Tx Power**	**ACAD**	**Adv Data**
00b	Chained data	X	X	C3	O	X	O	X	O
01b – 11b	RFU								

Table 2.7: Common Extended Advertising Payload Format fields permitted in the AUX_CHAIN_IND PDU

2.3.2 Scanning PDUs

The following advertising channel PDU types are called scanning PDUs.

- SCAN_REQ
- SCAN_RSP
- AUX_SCAN_REQ
- AUX_SCAN_RSP

The SCAN_REQ and AUX_SCAN_REQ PDUs are called scan request PDUs. The SCAN_RSP and AUX_SCAN_RSP PDUs are called scan response PDUs.

Where these PDUs are used to reply to a scannable advertisement, the PHY used for them shall be the same as the PHY used for the PDU that they reply to.

2.3.2.1 SCAN_REQ and AUX_SCAN_REQ

The SCAN_REQ and AUX_SCAN_REQ PDUs have the Payload as shown in Figure 2.10. The TxAdd in the advertising channel PDU header indicates whether the scanner's address in the ScanA field is public (TxAdd = 0) or random (TxAdd = 1). The RxAdd in the advertising channel PDU header 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 2.10: SCAN_REQ and AUX_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.

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

Note: These PDUs do not contain any Host Data.

2.3.2.2 SCAN_RSP

The SCAN_RSP PDU has a format as shown in Figure 2.11. The TxAdd in the advertising channel PDU header 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 2.11: 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.

2.3.2.3 AUX_SCAN_RSP

The AUX_SCAN_RSP PDU uses the Common Extended Advertising Payload Format described in Section 2.3.4.

The AdvMode field shall be set to 00b.

The Common Extended Advertising Payload Format fields permitted in the AUX_SCAN_RSP PDU are shown in Table 2.8.

Any auxiliary PDU shall be an AUX_CHAIN_IND PDU.

		Common Extended Advertising Payload Format fields							
Adv Mode	**Event Type**	**AdvA**	**TargetA**	**ADI**	**Aux Ptr**	**Sync Info**	**Tx Power**	**ACAD**	**Adv Data**
00b	Scan response	M	X	X	O	X	O	O	M
01b – 11b	RFU								

Table 2.8: Common Extended Advertising Payload Format fields permitted in the AUX_SCAN_RSP PDU

2.3.3 Initiating PDUs

The following advertising channel PDU Types are called initiating PDUs:

- CONNECT_IND
- AUX_CONNECT_REQ
- AUX_CONNECT_RSP

The CONNECT_IND and the AUX_CONNECT_REQ PDUs are sent by the Link Layer in the Initiating State and received by the Link Layer in the Advertising State.
The AUX_CONNECT_RSP PDU is sent by the Link Layer in the Advertising State and received by the Link Layer in the Initiating State.

The PHY used for these PDUs shall be the same as the PHY used for the PDU that they reply to.

2.3.3.1 CONNECT_IND and AUX_CONNECT_REQ

The CONNECT_IND and AUX_CONNECT_REQ PDUs have the Payload as shown in Figure 2.12. TxAdd in the advertising channel PDU header indicates whether the initiator's device address in the InitA field is public (TxAdd = 0) or random (TxAdd = 1). The RxAdd in the advertising channel PDU header indicates whether the advertiser's device address in the AdvA field is public (RxAdd = 0) or random (RxAdd = 1).

The ChSel field in the CONNECT_IND PDU header shall be set to 1 if the initiator supports the LE Channel Selection Algorithm #2 feature (see Section 4.5.8.3).The ChSel field in the AUX_CONNECT_REQ PDU header is Reserved for Future Use.

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

Figure 2.12: CONNECT_IND and AUX_CONNECT_REQ PDU payload

The format of the LLData field is shown in Figure 2.13.

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 2.13: LLData field structure in CONNECT_IND and AUX_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 seed for the random number generator shall be from a physical source of entropy and should have at least 20 bits of entropy.
- 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 advertising channel 38 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 2.9.

SCA	*masterSCA*
0	251 ppm to 500 ppm
1	151 ppm to 250 ppm

Table 2.9: SCA field encoding

SCA	*masterSCA*
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 2.9: SCA field encoding

2.3.3.2 AUX_CONNECT_RSP

The AUX_CONNECT_RSP PDU uses the Common Extended Advertising Payload Format described in Section 2.3.4.

The AdvMode field shall be set to 00b.

The Common Extended Advertising Payload Format fields permitted in the AUX_CONNECT_RSP PDU are shown in Table 2.10.

		Common Extended Advertising Payload Format fields							
Adv Mode	**Event Type**	**AdvA**	**TargetA**	**ADI**	**Aux Ptr**	**Sync Info**	**Tx Power**	**ACAD**	**Adv Data**
00b	Connection response	M	M	X	X	X	X	X	X
01b – 11b	RFU								

Table 2.10: Common Extended Advertising Payload Format fields permitted in the AUX_CONNECT_RSP PDU

2.3.4 Common Extended Advertising Payload Format

The following extended Advertising Channel PDUs share the same Advertising Channel PDU payload format, referred to in this specification as the “Common Extended Advertising Payload Format”:

- ADV_EXT_IND
- AUX_ADV_IND
- AUX_SCAN_RSP
- AUX_SYNC_IND
- AUX_CHAIN_IND
- AUX_CONNECT_RSP

The common extended advertising payload format is shown in Figure 2.14.

Payload			
Extended Header Length (6 bits)	AdvMode (2 bits)	Extended Header (0 - 63 octets)	AdvData (0 - 254 octets)

Figure 2.14: Common Extended Advertising Payload Format

The Extended Header Length is a value between 0 and 63 and indicates the size of the variable length Extended Header field.

The AdvMode field indicates the mode of the advertisement. The value of the AdvMode field shall be set as defined in Table 2.11.

Value	Mode	
00b	Non-connectable	Non-scannable
01b	Connectable	Non-scannable
10b	Non-connectable	Scannable
11b	Reserved for future use	

Table 2.11: AdvMode field encoding

AdvData may contain advertising data from the advertiser’s Host. The maximum size of AdvData depends on the size of the Extended Header. The size of the AdvData can be calculated by subtracting the length of the Extended Header plus one octet from the Length specified in the Advertising channel PDU Header.

The Extended Header field is a variable length header that is present if, and only if, the Extended Header Length field is non-zero. The format of the Extended Header is shown in Figure 2.15.

Extended Header								
Extended Header Flags (1 octet)	AdvA (6 octets)	TargetA (6 octets)	RFU (1 octet)	AdvData Info (ADI) (2 octets)	AuxPtr (3 octets)	SyncInfo (18 octets)	TxPower (1 octet)	ACAD (varies)

Figure 2.15: Extended Header

The Extended Header Flags bit field definitions are shown in Table 2.12.

Bit	Extended Header
0	AdvA
1	TargetA
2	Reserved for future use

Table 2.12: Extended Header Flags

Bit	Extended Header
3	AdvDataInfo (ADI)
4	AuxPtr
5	SyncInfo
6	TxPower
7	Reserved for future use

Table 2.12: Extended Header Flags

If a flag bit is set to 1, the corresponding Extended Header field is present; otherwise, the corresponding Extended Header field is not present. The Extended Header fields that are present are always in the same order as the flags in the Extended Header flags (i.e., the AdvA field is first if present, then the TargetA field if present, etc.).

Whether an Extended Header flag and corresponding Extended Header field is mandatory, optional, or reserved for future use is dependent on the Advertising Channel PDU in which the extended header is used.

2.3.4.1 AdvA field

When present, the AdvA field is six octets with the format shown in Figure 2.16.

AdvA
Advertising Address (6 octets)

Figure 2.16: AdvA field

The Advertising Address field contains the advertiser's device address. The TxAdd field of the Advertising Channel PDU Header applies to this value.

2.3.4.2 TargetA field

When present, the TargetA field is six octets with the format shown in Figure 2.17.

TargetA
Target Address (6 octets)

Figure 2.17: TargetA field

The Target Address field contains the scanner's or initiator's device address to which the advertisement is directed. The RxAdd field of the Advertising Channel PDU Header applies to this value.

2.3.4.3 RFU

<Field and section reserved for a future feature>

2.3.4.4 AdvDataInfo field

When present, the AdvDataInfo (ADI) field is two octets with the format shown in Figure 2.18.

AdvDataInfo	
Advertising Data ID (DID) (12 bits)	Advertising Set ID (SID) (4 bits)

Figure 2.18: AdvDataInfo field

The Advertising Set ID (SID) is set by the advertiser to distinguish between different advertising sets transmitted by this device.

The Advertising Data ID (DID) is set by the advertiser to indicate to the scanner whether it can assume that the data contents in the AdvData are a duplicate of the previous AdvData sent in an earlier packet.

2.3.4.5 AuxPtr field

When present, the AuxPtr field is three octets with the format shown in Figure 2.19.

AuxPtr				
Channel Index (6 bits)	CA (1 bits)	Offset Units (1 bits)	AUX Offset (13 bits)	AUX PHY (3 bits)

Figure 2.19: AuxPtr Field

The presence of the AuxPtr field indicates that some or all of the advertisement data is in a subsequent auxiliary packet. The contents of the AuxPtr field describe this packet.

The Channel Index field contains the secondary advertising channel index (see Section 1.4.1) used to transmit the auxiliary packet.

The Offset Units field indicates the units used by the Aux Offset Field. The value of the Offset Units field shall be set as defined in Table 2.13.

Value	**Units**
0	30 μs

Table 2.13: Offset Units field encoding

Value	Units
1	300 µs

Table 2.13: Offset Units field encoding

The Aux Offset field contains the time from the start of the packet containing the AuxPtr field to the approximate start of the auxiliary packet. The value of the AUX Offset field is in the unit of time indicated by the Offset Units field; the offset is determined by multiplying the value by the unit. The Aux Offset shall be at least the length of the packet plus T_MAFS (see Section 4.1.2). The Offset Units field shall be set to 0 if the Aux Offset is less than 245,700 µs.The auxiliary packet shall not start any earlier than the Aux Offset and shall start no later than the Aux Offset plus one Offset Unit. This allows the LL to round the Aux Offset to the Offset Unit.

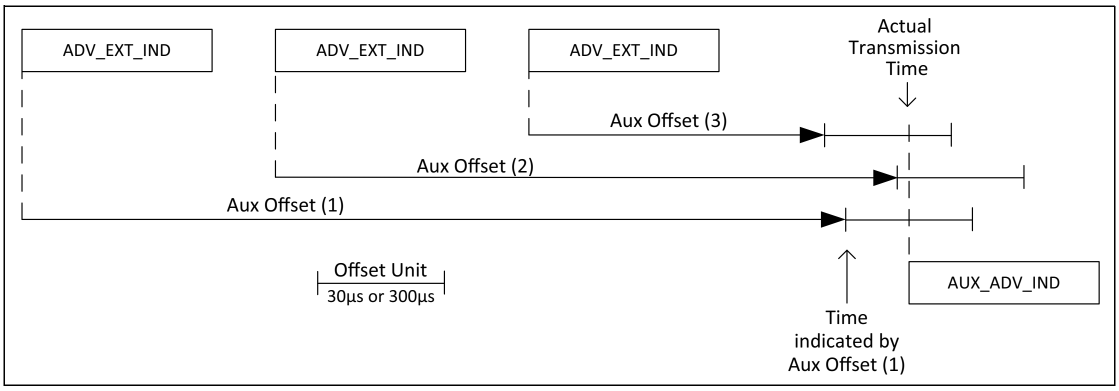


Figure 2.20: Aux Offset Transmission Window

The Aux PHY field indicates the PHY used to transmit the auxiliary packet. The value of the Aux PHY field shall be set as defined in Table 2.14.

Value	PHY used
000b	LE 1M
001b	LE 2M
010b	LE Coded
011b – 111b	Reserved for future use

Table 2.14: Aux PHY field encoding

The CA field contains the clock accuracy of the advertiser that will be used between the packet containing this data and the auxiliary packet. The value of the CA field shall be set as defined in Table 2.15.

CA Value	Advertiser’s Clock Accuracy
0	51 ppm to 500 ppm

Table 2.15: Clock Accuracy field encoding

CA Value	Advertiser's Clock Accuracy
1	0 ppm to 50 ppm

Table 2.15: Clock Accuracy field encoding

An AuxPtr field with an Aux Offset of zero is permitted and indicates that no auxiliary packet will be transmitted but the Host advertising data in the current PDU is incomplete (see Section 2.3.4.9); it shall be treated as equivalent to one referring to an AUX_CHAIN_IND PDU that is never received. The remaining fields shall contain valid values.

2.3.4.6 SyncInfo field

When present, the SyncInfo field is 18 octets with the format shown in Figure 2.21.

SyncInfo								
Sync Packet Offset (13 bits)	Offset Units (1 bit)	RFU (2 bits)	Interval (2 octets)	ChM (37 bits)	SCA (3 bits)	AA (4 octets)	CRCInit (3 octets)	Event Counter (2 octets)

Figure 2.21: SyncInfo field

The presence of the SyncInfo field indicates the presence of a periodic advertisement (using AUX_SYNC_IND PDUs). The contents of the SyncInfo field describe this periodic advertisement.

The Offset Units field indicates the units used by the Sync Packet Offset field. The value of the Offset Units field shall be set as defined in Table 2.16.

Value	Units
0	30 µs
1	300 µs

Table 2.16: Offset Units field encoding

The Sync Packet Offset field contains the time from the start of the AUX_ADV_IND packet containing the SyncInfo field to the start of the AUX_SYNC_IND packet. The value of the Sync Packet Offset field is in the unit of time indicated by the Offset Units field; the actual offset is determined by multiplying the value by the unit. The Offset Units field shall be set to 0 if the Sync Packet Offset is less than 245,700 µs. As illustrated in Figure 2.22, the AUX_ADV_IND packet containing the SyncInfo field shall start no later than the Sync Packet Offset and no earlier than the Sync Packet Offset plus one Offset unit prior to the start of the AUX_SYNC_IND.

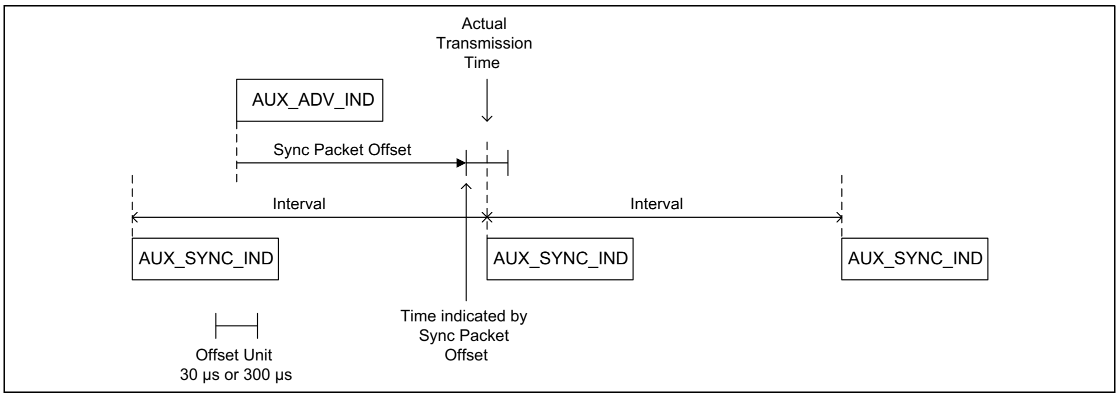


Figure 2.22: Sync Packet Offset Transmission Window

A value of 0 for the Sync Packet Offset indicates that the time to the next AUX_SYNC_IND packet is greater than can be represented.

The Interval field contains the time in 1.25 ms units from the start of one packet of the periodic advertisement to the start of the next packet. The value shall not be less than 6 (7.5 ms).

The ChM field contains the channel map indicating *Used* and *Unused* secondary advertising 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 channel index 0 and the bit in position 36 represents 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 AA, CRCInit, and SCA fields have the same meaning as the corresponding fields in the CONNECT_IND PDU (see Section 2.3.3.1).

The Event Counter field contains the value of *paEventCounter* (see Section 4.4.2.1) that applies to the AUX_SYNC_IND packet that this SyncInfo field describes.

2.3.4.7 TxPower field

When present, the TxPower is one octet with the format shown in Figure 2.23.

TxPower
Tx Power Level (1 octet)

Figure 2.23: TxPower field

The Tx Power Level field is the same value defined for the Tx Power Advertising Data type defined in Bluetooth Core Specification Supplement (CSS).

2.3.4.8 ACAD field

The remainder of the extended header forms the Additional Controller Advertising Data (ACAD) field. The length of this field is the Extended Header length minus the sum of the size of the extended header flags (1 octet) and those fields indicated by the flags as present. ACAD cannot be fragmented across multiple advertising data PDUs; it shall always fit inside a single advertising data PDU.

The ACAD field holds data from the advertiser's Controller or intended to be used by the recipient's Controller. It uses the same format as the AdvData field in various Advertising Channel PDUs. This format is described in [Vol 3] Part C, Section 11.

The ACAD type formats and meanings are defined in [CSS], Part A, Section 1. The ACAD type identifier values are defined in the Assigned Numbers document.

2.3.4.9 Host Advertising Data

The portion of the PDU after the extended header forms the AdvData field. The length of this field is specified in Section 2.3.4.

The AdvData field holds data from the advertiser's Host. The format of this data is described in [Vol 3] Part C, Section 11. If the Host does not provide any data, the AdvData field shall be omitted but, for all other purposes in this Part, this shall be treated as if the Host had provided data.

The Controller may support fragmentation of Host Advertising Data. The total amount of Host Advertising Data before fragmentation shall not exceed 1650 octets. When the Link Layer fragments the Host advertising data, the number of fragments and the size of each fragment are chosen by the Controller. The Controller should minimize the number of fragments to ensure more reliability in delivering the entire Host advertising data. The Host may indicate a preference whether the Controller should fragment the Host advertising data, but the Controller may ignore the preference. If the amount of advertising or scan response data to be sent in an extended advertising or scanning PDU plus the Extended Header Length, AdvMode, and Extended Header exceed the maximum Advertising Channel PDU payload (255 octets), the Link Layer shall fragment the Host advertising data.

The Link Layer shall place the multiple fragments in the AdvData field of different PDUs. When Host advertising data is fragmented the first fragment shall be placed in the AUX_ADV_IND, AUX_SYNC_IND or AUX_SCAN_RSP PDU while subsequent fragments shall be placed in AUX_CHAIN_IND PDUs. Each AUX_CHAIN_IND PDU is the auxiliary PDU of the PDU containing the previous fragment; the AUX_CHAIN_IND PDU holding the last fragment shall not have an auxiliary PDU.

If the Link Layer has fragmented the Host advertising data but is subsequently unable to transmit all the fragments, the last fragment that it is able to transmit should contain an AuxPtr field with an Aux Offset of zero so that scanners are aware that the data has been truncated.

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 2.24.

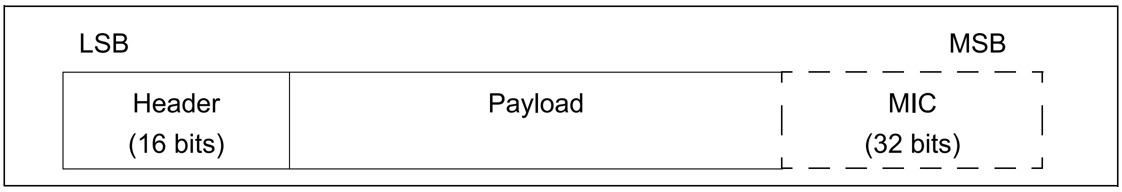


Figure 2.24: Data Channel PDU

The Header field of the Data Channel PDU is as shown in Figure 2.25.

Header					
LLID (2 bits)	NESN (1 bit)	SN (1 bit)	MD (1 bit)	RFU (3 bits)	Length (8 bits)

Figure 2.25: Data channel PDU header

The 16 bit Header field consists of 5 fields that are specified in Table 2.17.

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.

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 255 octets. The Payload field shall be less than or equal to 251 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 for future use 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 2.17: 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 00000000b, 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 00000000b.

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 2.26.

Payload	
Opcode (1 octet)	CtrData (0 – 26 octets)

Figure 2.26: LL control PDU payload

An LL Control PDU shall not have the Length field set to 00000000b. 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 2.18.

The CtrData field in the LL Control PDU is specified by the Opcode field and is defined in the following subsections.

Except where explicitly stated otherwise, all fields within the CtrData field in an LL Control PDU that hold an integer shall be interpreted as unsigned.

Opcode	**Control PDU Name**
0x00	LL_CONNECTION_UPDATE_IND
0x01	LL_CHANNEL_MAP_IND
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

Table 2.18: LL Control PDU Opcodes

Opcode	Control PDU Name
0x0D	LL_REJECT_IND
0x0E	LL_SLAVE_FEATURE_REQ
0x0F	LL_CONNECTION_PARAM_REQ
0x10	LL_CONNECTION_PARAM_RSP
0x11	LL_REJECT_EXT_IND
0x12	LL_PING_REQ
0x13	LL_PING_RSP
0x14	LL_LENGTH_REQ
0x15	LL_LENGTH_RSP
0x16	LL_PHY_REQ
0x17	LL_PHY_RSP
0x18	LL_PHY_UPDATE_IND
0x19	LL_MIN_USED_CHANNELS_IND
All other values	Reserved for Future Use

Table 2.18: 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 UnknownType field of the LL_UNKNOWN_RSP PDU shall be set to the value of the invalid Opcode.

2.4.2.1 LL_CONNECTION_UPDATE_IND

The format of the CtrData field is as shown in Figure 2.27.

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

Figure 2.27: CtrData field of the LL_CONNECTION_UPDATE_IND PDU

The LL_CONNECTION_UPDATE_IND 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 4.5.1, 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 4.5.2, in the following manner: *connSupervisionTimeout* = Timeout * 10 ms.
- The Instant field shall be set to indicate the instant described in Section 5.1.1.

2.4.2.2 LL_CHANNEL_MAP_IND

The format of the CtrData field is shown in Figure 2.28.

CtrData	
ChM (5 octets)	Instant (2 octets)

Figure 2.28: CtrData field of the LL_CHANNEL_MAP_IND PDU

The LL_CHANNEL_MAP_IND 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_IND PDU (see Section 2.3.3.1).
- The Instant field shall be set to indicate the instant described in Section 5.1.2.

2.4.2.3 LL_TERMINATE_IND

The format of the CtrData field is shown in Figure 2.29.

CtrData
ErrorCode (1 octet)

Figure 2.29: CtrData field of the LL_TERMINATE_IND PDU

The LL_TERMINATE_IND CtrData consists of one field:

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

2.4.2.4 LL_ENC_REQ

The format of the CtrData field is shown in Figure 2.30.

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

Figure 2.30: 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 diversifier.
- 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 2.31.

CtrData	
SKDs (8 octets)	IVs (4 octets)

Figure 2.31: 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 diversifier.
- 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 2.32.

CtrData
UnknownType (1 octet)

Figure 2.32: 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 2.33.

CtrData
FeatureSet (8 octets)

Figure 2.33: CtrData field of the LL_FEATURE_REQ PDU

The LL_FEATURE_REQ CtrData consists of one field:

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

2.4.2.10 LL_FEATURE_RSP

The format of the CtrData field is shown in Figure 2.34.

CtrData
FeatureSet (8 octets)

Figure 2.34: CtrData field of the LL_FEATURE_RSP PDU

The LL_FEATURE_RSP CtrData consists of one field:

- FeatureSet[0] shall contain a set of features supported by the Link Layers of both the master and slave.
- FeatureSet[1-7] shall contain a set of features supported by the Link Layer that transmits this PDU.

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 2.35.

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

Figure 2.35: 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 Link Layer 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 2.36.

CtrData
ErrorCode (1 octet)

Figure 2.36: CtrData field of the LL_ REJECT_IND

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

2.4.2.15 LL_SLAVE_FEATURE_REQ

The format of the CtrData field is shown in Figure 2.37.

CtrData
FeatureSet (8 octets)

Figure 2.37: CtrData field of the LL_SLAVE_FEATURE_REQ PDU

The LL_SLAVE_FEATURE_REQ CtrData consists of one field:

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

2.4.2.16 LL_CONNECTION_PARAM_REQ

The format of the CtrData field is shown in Figure 2.38.

CtrData											
Interval_Min (2 octets)	Interval_Max (2 octets)	Latency (2 octets)	Timeout (2 octets)	PreferredPeriodicity (1 octet)	ReferenceConnEventCount (2 octets)	Offset0 (2 octets)	Offset1 (2 octets)	Offset2 (2 octets)	Offset3 (2 octets)	Offset4 (2 octets)	Offset5 (2 octets)

Figure 2.38: CtrData field of the LL_CONNECTION_PARAM_REQ PDU

The LL_CONNECTION_PARAM_REQ CtrData consists of 12 fields:

- The Interval_Min field shall be set to indicate the minimum value of *connInterval*, as defined in Section 4.5.1, in the following manner: *connInterval* = Interval_Min * 1.25 ms.
- The Interval_Max field shall be set to indicate the maximum value of *connInterval*, as defined in Section 4.5.1, in the following manner: *connInterval* = Interval_Max * 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. Latency is in units of number of connection events.
- The Timeout field shall be set to indicate the *connSupervisionTimeout* value, as defined by Section 4.5.2, in the following manner: *connSupervisionTimeout* = Timeout * 10 ms.
- The PreferredPeriodicity field shall be set to indicate a value the *connInterval* is preferred to be a multiple of. PreferredPeriodicity is in units of 1.25 ms. E.g. if the PreferredPeriodicity is set to 100, it implies that *connInterval* is preferred to be any multiple of 125 ms. A value of zero means not valid. The PreferredPeriodicity shall be less than or equal to Interval_Max.
- The ReferenceConnEventCount field shall be set to indicate the value of the *connEventCounter* relative to which all the valid Offset0 to Offset5 fields have been calculated. The ReferenceConnEventCount field shall have a value in the range of 0 to 65535. Note: The ReferenceConnEventCount field is independent of the Instant field in the LL_CONNECTION_UPDATE_IND PDU.
- The Offset0, Offset1, Offset2, Offset3, Offset4, and Offset5 fields shall be set to indicate the possible values of the position of the anchor points of the LE connection with the updated connection parameters relative to the

ReferenceConnEventCount. The Offset0 to Offset5 fields are in units of 1.25 ms and are in decreasing order of preference; that is, Offset0 is the most preferred value, followed by Offset1, and so on. Offset0 to Offset5 shall be less than Interval_Max. A value of 0xFFFF means not valid. Valid Offset0 to Offset5 fields shall contain unique values. Valid fields shall always be before invalid fields.

2.4.2.17 LL_CONNECTION_PARAM_RSP

The format of the LL_CONNECTION_PARAM_RSP PDU is identical to the format of the LL_CONNECTION_PARAM_REQ PDU (see Section 2.4.2.16).

2.4.2.18 LL_REJECT_EXT_IND

The format of the CtrData field is shown in Figure 2.39.

CtrData	
RejectOpcode (1 octet)	ErrorCode (1 octet)

Figure 2.39: CtrData field of the LL_REJECT_EXT_IND PDU

The LL_REJECT_EXT_IND CtrData consists of two fields:

- RejectOpcode shall contain the Opcode field value of the LL Control PDU being rejected.
- ErrorCode shall contain the reason the LL Control PDU was being rejected. See [Vol 2] Part D, Error Codes for a list of error codes and descriptions.

This PDU shall be issued only when the remote Link Layer supports the Extended Reject Indication Link Layer feature (Section 4.6). Otherwise, the LL_REJECT_IND PDU (Section 2.4.2.14) shall be issued instead.

2.4.2.19 LL_PING_REQ

The LL_PING_REQ PDU does not have a CtrData field.

2.4.2.20 LL_PING_RSP

The LL_PING_RSP PDU does not have a CtrData field.

2.4.2.21 LL_LENGTH_REQ and LL_LENGTH_RSP

The format of the CtrData field for both the LL_LENGTH_REQ and LL_LENGTH_RSP PDUs is shown in Figure 2.40.

CtrData			
MaxRxOctets (2 octets)	MaxRxTime (2 octets)	MaxTxOctets (2 octets)	MaxTxTime (2 octets)

Figure 2.40: CtrData field of the LL_LENGTH_REQ and LL_LENGTH_RSP PDUs

The LL_LENGTH_REQ and LL_LENGTH_RSP CtrData consists of four fields:

- MaxRxOctets shall be set to the sender's *connMaxRxOctets* value, as defined in Section 4.5.10. The MaxRxOctets field shall have a value not less than 27 octets.
- MaxRxTime shall be set to the sender's *connMaxRxTime* value, as defined in Section 4.5.10. The MaxRxTime field shall have a value not less than 328 microseconds.
- MaxTxOctets shall be set to the sender's *connMaxTxOctets* value, as defined in Section 4.5.10. The MaxTxOctets field shall have a value not less than 27 octets.
- MaxTxTime shall be set to the sender's *connMaxTxTime* value, as defined in Section 4.5.10. The MaxTxTime field shall have a value not less than 328 microseconds.

2.4.2.22 LL_PHY_REQ and LL_PHY_RSP

The format of the CtrData field for both the LL_PHY_REQ and LL_PHY_RSP PDUs is shown in Figure 2.41.

CtrData	
TX_PHYS (1 octet)	RX_PHYS (1 octet)

Figure 2.41: CtrData field of the LL_PHY_REQ and LL_PHY_RSP PDUs

The LL_PHY_REQ and LL_PHY_RSP CtrData consists of two fields:

- TX_PHYS shall be set to indicate the transmitter PHYs that the sender prefers to use.
- RX_PHYS shall be set to indicate the receiver PHYs that the sender prefers to use.

These fields each consist of 8 bits as specified in Table 2.19. At least one bit in each field shall be set to 1.

Bit number	Meaning
0	Sender prefers to use the LE 1M PHY (possibly among others)
1	Sender prefers to use the LE 2M PHY (possibly among others)
2	Sender prefers to use the LE Coded PHY (possibly among others)
3–7	Reserved for future use

Table 2.19: PHY field bit meanings

2.4.2.23 LL_PHY_UPDATE_IND

The format of the CtrData field is shown in Figure 2.42.

CtrlData		
M_TO_S_PHY (1 octet)	S_TO_M_PHY (1 octet)	Instant (2 octets)

Figure 2.42: CtrData field of the LL_PHY_UPDATE_IND PDU

The LL_PHY_UPDATE_IND CtrData consists of three fields:

- M_TO_S_PHY shall be set to indicate the PHY that shall be used for packets sent from the master to the slave. S_TO_M_PHY shall be set to indicate the PHY that shall be used for packets sent from the slave to the master. These fields each consist of 8 bits as specified in Table 2.20. If a PHY is changing, exactly one bit shall be set to the value 1 in the corresponding field; if a PHY is remaining unchanged, then the corresponding field shall be set to the value 0.
- Instant shall be set to indicate the instant described in Section 5.1.10.

If both the M_TO_S_PHY and S_TO_M_PHY fields are zero then there is no Instant and the Instant field is reserved for future use.

Bit Number	Meaning
0	The LE 1M PHY shall be used
1	The LE 2M PHY shall be used
2	The LE Coded PHY shall be used
3–7	Reserved for future use

Table 2.20: PHY field bit meanings

2.4.2.24 LL_MIN_USED_CHANNELS_IND

The format of the CtrData field is as shown in Figure 2.43.

CtrData	
PHYS (1 octet)	MinUsedChannels (1 octet)

Figure 2.43: CtrData field of the LL_MIN_USED_CHANNELS_IND PDU

The LL_MIN_USED_CHANNELS_IND consists of two fields:

- The PHYS field shall be set to the PHY(s) for which the slave has a minimum number of used channels requirement. The PHYS field consists of 8 bits as specified in Table 2.21. At least one bit in the field shall be set to 1.
- The MinUsedChannels field contains the minimum number of channels to be used on the specified PHY. The MinUsedChannels field shall have a value in the range of 2 to 37 channels.

Bit Number	Meaning
0	LE 1M PHY
1	LE 2M PHY
2	LE Coded PHY
3-7	Reserved for Future Use

Table 2.21: PHY field bit meanings

3 BIT STREAM PROCESSING

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

Figure 3.1 shows the bit stream processing for PDUs on the LE Uncoded PHYs.

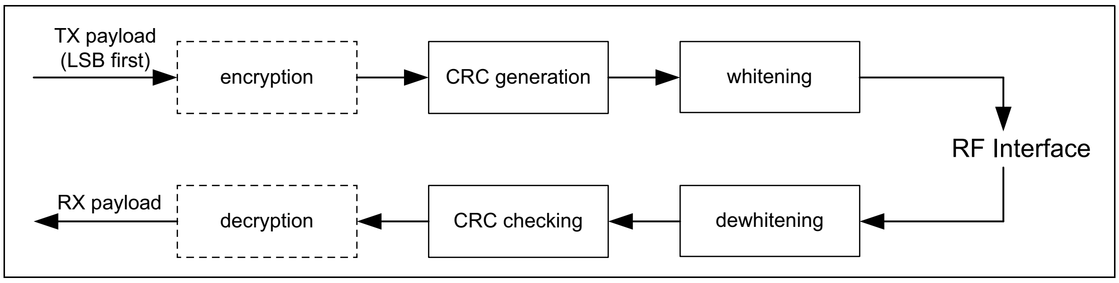


Figure 3.1: Payload bit processes for the LE Uncoded PHYs

Figure 3.2 shows the bit stream processing for PDUs on the LE Coded PHYs.

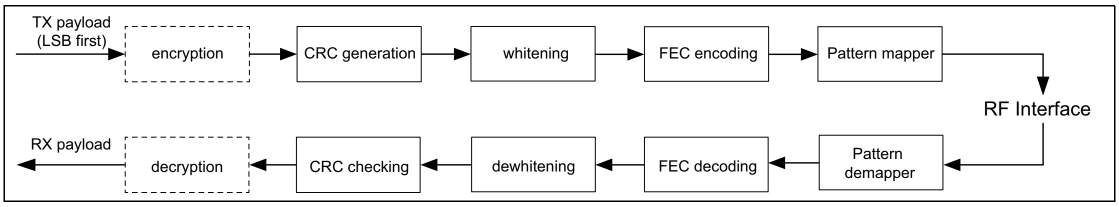


Figure 3.2: Bit stream processing for the LE Coded PHYs

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_IND PDU. For the

AUX_SYNC_IND PDU and its subordinate set, the shift register shall be preset with the CRCInit value set in the SyncInfo field (see Section 2.3.4.6) contained in the AUX_ADV_IND PDU that describes the periodic advertising. For all other Advertising Channel PDUs, 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 3.3 shows an example linear feedback shift register (LFSR) to generate the CRC.

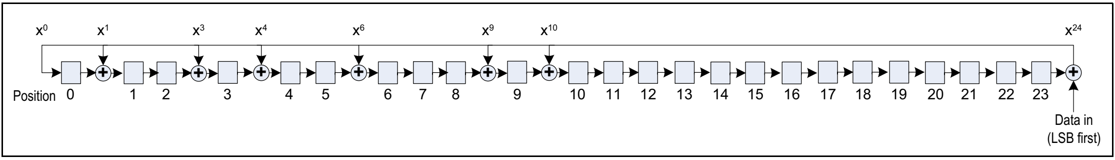

x0
x1
x3
x4
x6
x9
x10
x24
Position 0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23
Data in (LSB first)

Figure 3.3: 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 packets 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 3.4 shows an example linear feedback shift register (LFSR) to generate data whitening.

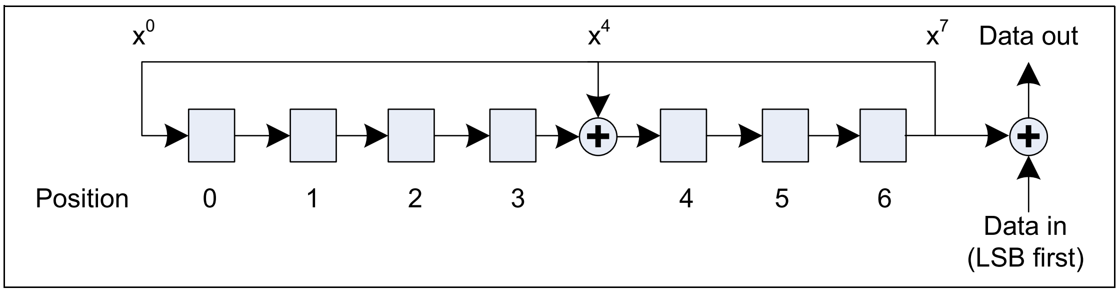


Figure 3.4: The LFSR circuit to generate data whitening

3.3 CODING

Coding only applies to the LE Coded PHY.

Coding consists of two processes. Data is first coded by the Forward Error Correction (FEC) convolutional encoder as defined in Section 3.3.1 and then spread by the pattern mapper as defined in Section 3.3.2.

3.3.1 Forward Error Correction Encoder

The convolutional FEC encoder uses a non-systematic, non-recursive rate ½ code with constraint length K=4. The generator polynomials are:

$G_0(x) = 1 + x + x^2 + x^3$

$G_1(x) = 1 + x^2 + x^3$

The bit coming from generator polynomial G_0 (a_0) is transmitted first; the bit coming from generator polynomial G_1 (a_1) is transmitted second.

The initial state of the convolutional FEC encoder is set to all zeros. An input sequence of three consecutive zeros always brings the convolutional FEC encoder back to its original state. This sequence is known as the termination sequence.

Figure 3.5 illustrates operation of the convolutional FEC encoder. Squares represent bit storage operations and circles represent mod 2 binary additions.

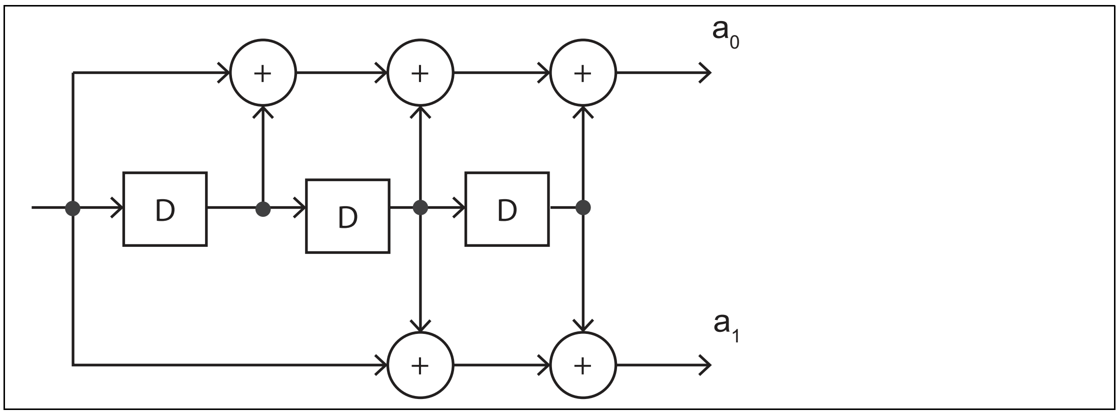


Figure 3.5: Convolutional Forward Error Correction Encoder

3.3.2 Pattern Mapper

The pattern mapper converts each bit from the convolutional FEC encoder into P symbols, where the value of P depends on the coding scheme in use, according to Table 3.1 (the entries in the table are in transmission order):

Input bit from the convolutional FEC encoder	**Output sequence when P=1 (used by S=2)**	**Output sequence when P=4 (used by S=8)**
0	0	0011
1	1	1100

Table 3.1: Pattern Mapper inputs and outputs

4 AIR INTERFACE PROTOCOL

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

4.1 FRAME SPACE

4.1.1 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.

4.1.2 Minimum AUX Frame Space

The minimum time interval between a packet containing an AuxPtr and the auxiliary packet it indicates is called the Minimum AUX Frame Space. It is defined as the minimum time from the end of the last bit of the packet containing the AuxPtr to the start of the auxiliary packet. The Minimum AUX Frame Space is designated "T_MAFS" and shall be 300 μs.

Figure 4.1 illustrates an example where the Minimum AUX Frame Space applies.

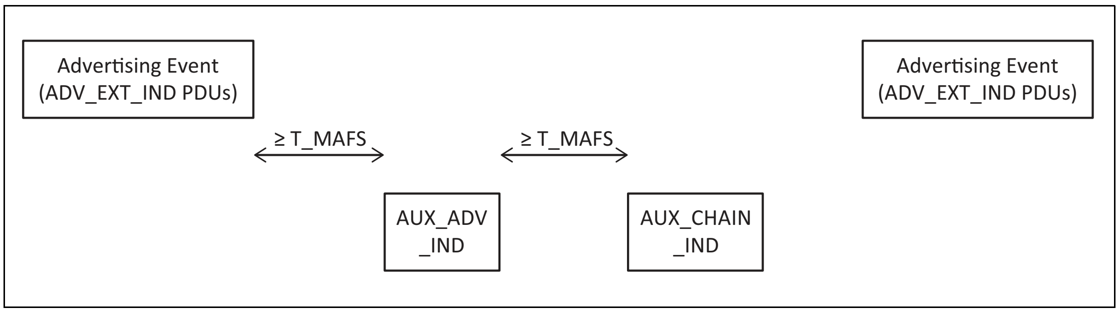

Advertising Event (ADV_EXT_IND PDUs)
≥ T_MAFS
≥ T_MAFS
AUX_ADV _IND
AUX_CHAIN _IND
Advertising Event (ADV_EXT_IND PDUs)

Figure 4.1: Example where the Minimum AUX Frame Space applies

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.

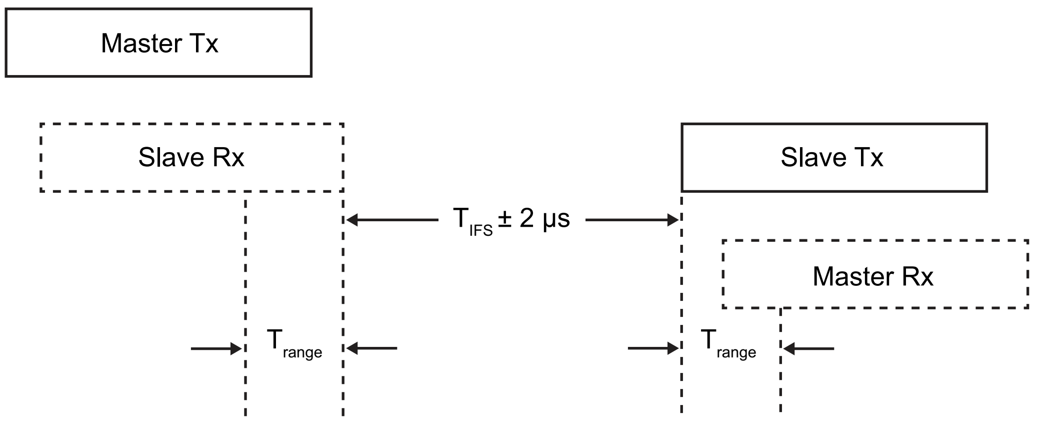
Note: This means that a 1 s connection interval with a total ±1000ppm sleep clock accuracy will give a window widening either side of the anchor point of 1ms plus 16us, assuming that the slave Controller was using its sleep clock for almost the complete connection interval.

4.2.3 Range Delay

Where two devices are more than a few meters apart the time taken for a signal to propagate between them will be significant compared with the Active Clock Accuracy defined in Section 4.2.1. When a device is listening for a packet that might be up to D meters away, it should listen for an extra 2D * 4 ns after the nominal latest time (e.g. T_IFS + 2 μs) that the packet would have been transmitted.

(1/c ~= 3.3 * refractive index ns/m, so 4 ns gives a conservative allowance.)

Figure 4.2 shows the range delays relative to a master packet transmission.



Relative to the Master Tx:

Slave Tx = (T_{IFS} ± 2 μs) + T_{range}
Master Rx = (T_{IFS} ± 2 μs) + (2*T_{range})

Figure 4.2: Range delays relative to a master packet transmission

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). There is also a special device address type "anonymous"; an entry with this type matches all advertisements sent with no address. 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 non-connectable and non-scannable directed advertising events, scannable directed advertising events, and connectable directed advertising events 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 per advertising set 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. A connectable directed advertising packet not containing the scanner's device address shall be ignored.
- The Link Layer shall process all advertising packets (i.e., the White List is not used). A connectable directed advertising packet not containing the scanner's device address shall be ignored. This is the default on reset.

If the Link Layer supports the Extended Scanner Filter policies, then the following modes shall also be supported:

- The Link Layer shall process advertising packets only from devices in the White List. A connectable directed advertising packet shall not be ignored if the TargetA is the scanner's device address or a resolvable private address.
- The Link Layer shall process all advertising packets (i.e., the White List is not used). A connectable directed advertising packet shall not be ignored if the TargetA is the scanner's device address or a resolvable private address.

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 primary advertising channel indices. The advertising event shall be closed after one advertising PDU has been sent on each of the used primary advertising channel indices (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 and scannable undirected event
- a connectable undirected event
- a connectable directed event
- a non-connectable and non-scannable undirected event
- a non-connectable and non-scannable directed event
- a scannable undirected event
- a scannable directed event

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

The advertising event type determines the allowable response PDUs. Table 4.1 specifies the allowable responses for each advertising event.

Connectable and scannable undirected, connectable undirected, and connectable directed events are collectively referred to as connectable events;

the remaining events (non-connectable and non-scannable undirected, non-connectable and non-scannable directed, scannable undirected, and scannable directed) are collectively referred to as non-connectable events. Connectable and scannable undirected, scannable undirected, and scannable directed events are collectively referred to as scannable events; the remaining events (non-connectable and non-scannable undirected, non-connectable and non-scannable directed, connectable undirected, and connectable directed) are collectively referred to as non-scannable events.

		Allowable response PDUs			
Advertising Event Type	**Type of PDU being responded to**	**SCAN _REQ**[1]	**CONNECT _IND**[1]	**AUX_SCAN _REQ**	**AUX _CONNECT _REQ**
Connectable and Scannable Undirected Event	ADV_IND	YES	YES	NO	NO
Connectable Undirected Event	ADV_EXT_IND	NO	NO	NO	NO
	AUX_ADV_IND	NO	NO	NO	YES
Connectable Directed Event	ADV_DIRECT_IND	NO	YES[2]	NO	NO
	ADV_EXT_IND	NO	NO	NO	NO
	AUX_ADV_IND	NO	NO	NO	YES[2]
Non-Connectable and Non-Scannable Undirected Event	ADV_NONCONN_IND	NO	NO	NO	NO
	ADV_EXT_IND	NO	NO	NO	NO
	AUX_ADV_IND	NO	NO	NO	NO
Non-Connectable and Non-Scannable Directed Event	ADV_EXT_IND	NO	NO	NO	NO
	AUX_ADV_IND	NO	NO	NO	NO
Scannable Undirected Event	ADV_SCAN_IND	YES	NO	NO	NO
	ADV_EXT_IND	NO	NO	NO	NO
	AUX_ADV_IND	NO	NO	YES	NO
Scannable Directed Event	ADV_EXT_IND	NO	NO	NO	NO
	AUX_ADV_IND	NO	NO	YES[3]	NO

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

1. Not permitted on the LE Coded PHY.
2. Initiators other than the correctly addressed initiator shall not respond.
3. Scanners other than the correctly addressed scanner shall not 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 primary advertising channel index or close the advertising event.

4.4.2.1 Advertising Channel Index Selection

Advertising events use three predefined primary advertising channels. Primary advertising channel indices are either used or unused.

For AUX_ADV_IND and AUX_CHAIN_IND PDUs, the secondary advertising channel index used in the Channel Index subfield of the AuxPtr field is implementation specific. It is recommended that sufficient channel diversity is used to avoid collisions.

Each periodic advertisement shall have a 16-bit event counter (*paEventCounter*). The initial value of this counter is implementation specific. The counter shall be incremented by one for each AUX_SYNC_IND PDU; the *paEventCounter* shall wrap from 0xFFFF to 0x0000. AUX_SYNC_IND PDUs shall use the Channel Selection Algorithm #2 (see Section 4.5.8.3) with this event counter.

The Link Layer shall use the primary and secondary advertising channel indices as specified by the Host, and the used primary and secondary advertising channel indices shall take effect when the Advertising State is entered. The Link Layer need not use all the secondary channels that the Host has marked as "unknown".

4.4.2.2 Advertising Events

Advertising events are defined as one or more advertising PDUs sent on the primary advertising channel beginning with the first used advertising channel index and ending with the last used advertising channel index. The advertising event can be closed early after a CONNECT_IND is received or when a SCAN_RSP is sent. The time between advertising events is the advertising interval.

Advertising packets sent on the secondary advertising channel are not part of the advertising event. Advertising events that use the ADV_EXT_IND PDU may also be part of an extended advertising event. All ADV_EXT_IND PDUs containing an AuxPtr field in the same advertising event shall point to the same AUX_ADV_IND packet.

4.4.2.2.1 Advertising Interval

For all undirected advertising events or connectable directed advertising events used in a low duty cycle mode, the time between the start of two consecutive advertising events (*T_advEvent*) for the same advertising data set (see Section 4.4.2.10) 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 10,485.759375 s.

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 4.3, the advertising events are perturbed in time using the advDelay.

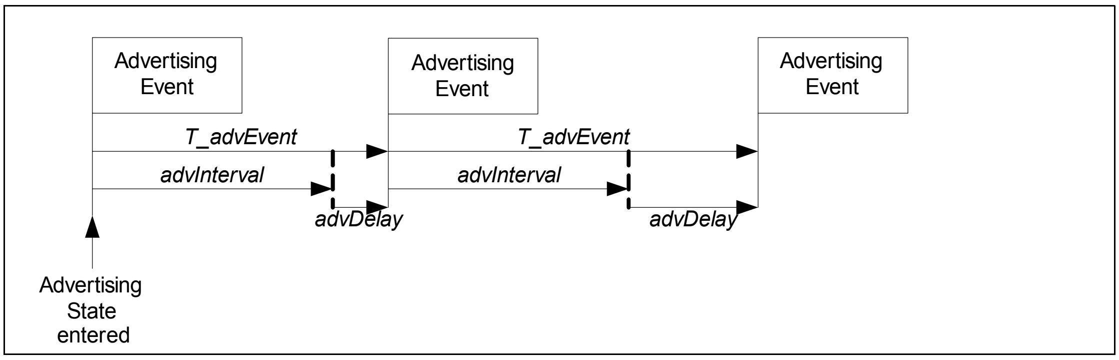


Figure 4.3: Advertising events perturbed in time using advDelay

4.4.2.2.2 Extended Advertising Event

An extended advertising event begins at the start of an advertising event and consists of the PDUs in that advertising event plus their subordinate sets. The extended advertising event ends with the last such PDU.

Multiple extended advertising events may overlap with each other. This can occur when ADV_EXT_IND PDUs containing an AuxPtr field in multiple advertising events point to the same AUX_ADV_IND packet, or when a different advertising event is interposed between the ADV_EXT_IND PDUs and the AUX_ADV_IND PDU.

T_advEvent, advInterval and advDelay have the same meaning as in Section 4.4.2.2.1.

Figure 4.4 illustrates an example of overlapping extended advertising events.

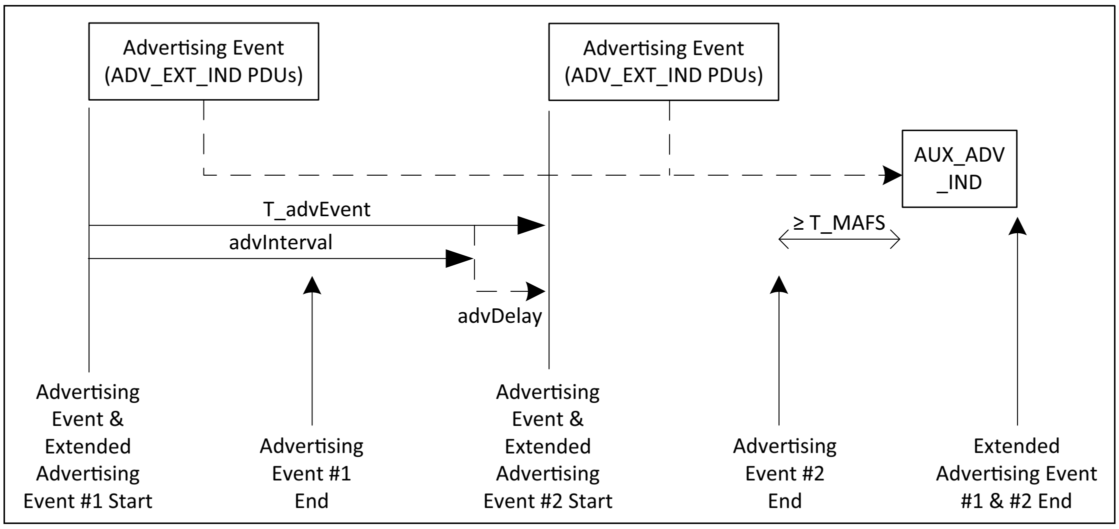

Advertising Event (ADV_EXT_IND PDUs)
Advertising Event (ADV_EXT_IND PDUs)
AUX_ADV _IND
T_advEvent
advInterval
advDelay
≥ T_MAFS
Advertising Event & Extended Advertising Event #1 Start
Advertising Event #1 End
Advertising Event & Extended Advertising Event #2 Start
Advertising Event #2 End
Extended Advertising Event #1 & #2 End

Figure 4.4: Example of overlapping extended advertising events

An auxiliary advertising segment starts with the first AUX_ADV_IND PDU in an extended advertising event and ends at the end of the extended advertising event (an auxiliary advertising segment can belong to more than one extended advertising event). Two auxiliary advertising segments for the same advertising set shall not overlap each other.

Figure 4.5 illustrates an example of auxiliary advertising segments belonging to multiple extended advertising events.

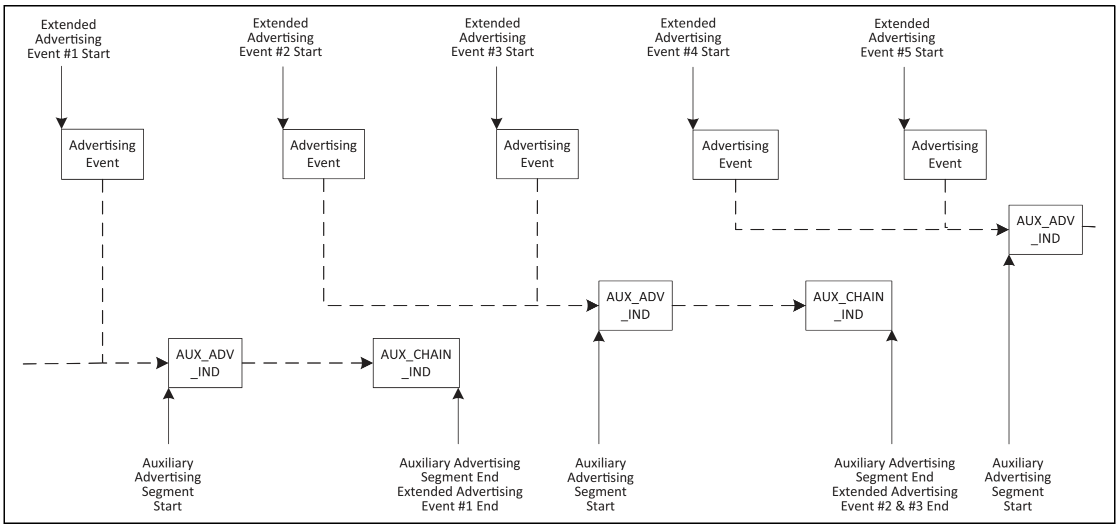

Extended Advertising Event #1 Start
Extended Advertising Event #2 Start
Extended Advertising Event #3 Start
Extended Advertising Event #4 Start
Extended Advertising Event #5 Start
Advertising Event
Advertising Event
Advertising Event
Advertising Event
Advertising Event
AUX_ADV _IND
AUX_ADV _IND
AUX_CHAIN _IND
AUX_ADV _IND
AUX_CHAIN _IND
Auxiliary Advertising Segment Start
Auxiliary Advertising Segment End Extended Advertising Event #1 End
Auxiliary Advertising Segment Start
Auxiliary Advertising Segment End Extended Advertising Event #2 & #3 End
Auxiliary Advertising Segment Start

Figure 4.5: Example of Auxiliary Advertising Segments

An advertiser should not space PDUs within the auxiliary advertising set so that two of them would be within the same receive window. If an advertiser transmits a PDU with an AuxPtr field containing an offset of T milliseconds, then it should not start to transmit any other packet on the same RF channel as the auxiliary packet within 2.5*T microseconds of the start of auxiliary packet of the original PDU.

4.4.2.2.3 Periodic Advertising Events

The Periodic Advertising Interval is the interval between the start of two AUX_SYNC_IND PDUs from the same advertising set. The Periodic Advertising Interval shall be an integer multiple of 1.25 ms in the range of 7.5 ms to 81.91875 s.

A periodic advertising event consists of an AUX_SYNC_IND PDU and its subordinate set.

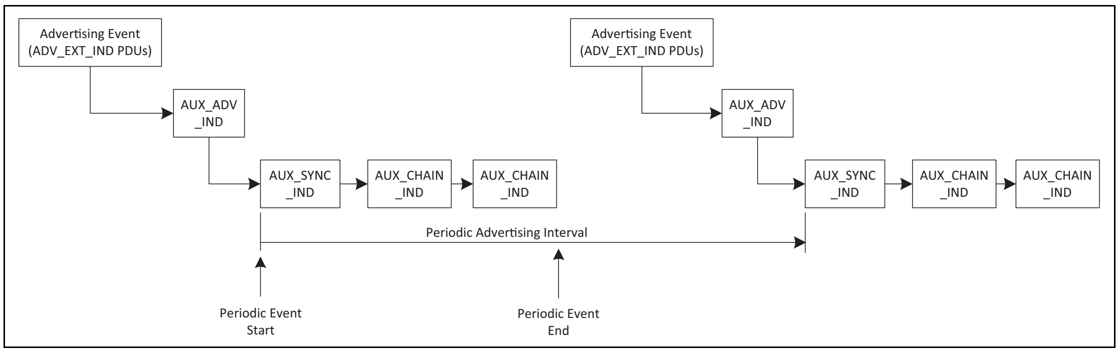


Figure 4.6: Example of Periodic Advertising Events from the same Advertising Set

Two periodic advertising events for the same advertising set shall not overlap each other. The periodic advertising interval shall not change while the periodic advertising is enabled.

4.4.2.3 Connectable and Scannable Undirected Event Type

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

The connectable and scannable 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_IND PDU) to request the Link Layer to enter the Connection State.

The Link Layer shall listen on the same primary 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 primary 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 primary advertising channel index to send another ADV_IND PDU, or close the advertising event.

If the advertiser receives a CONNECT_IND 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_IND PDU, the advertiser shall either move to the next used primary 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 event shall be closed within the advertising interval.

An illustration of an advertising event using all the primary advertising channel indices and in which no SCAN_REQ or CONNECT_IND PDUs are received is shown in Figure 4.7.

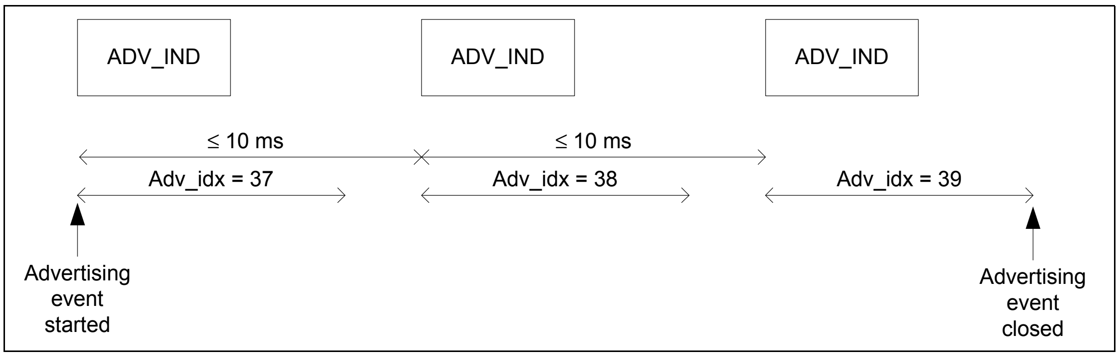


Figure 4.7: Connectable and scannable undirected advertising event with only advertising PDUs

Two illustrations of advertising events using all the primary advertising channel indices 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.

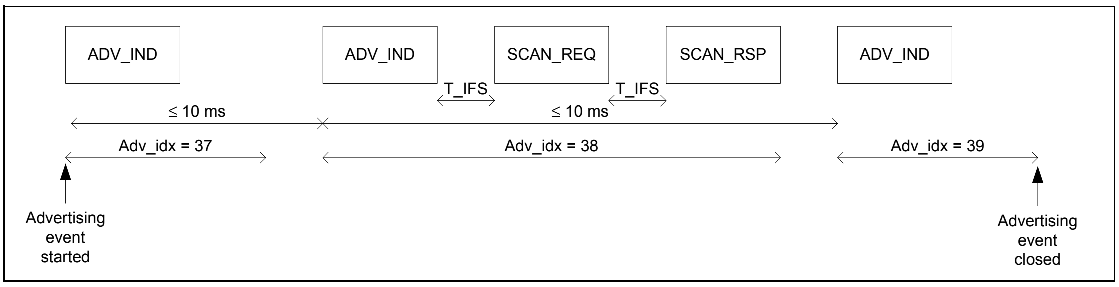


Figure 4.8: Connectable and scannable undirected advertising event with SCAN_REQ and SCAN_RSP PDUs in the middle of an advertising event

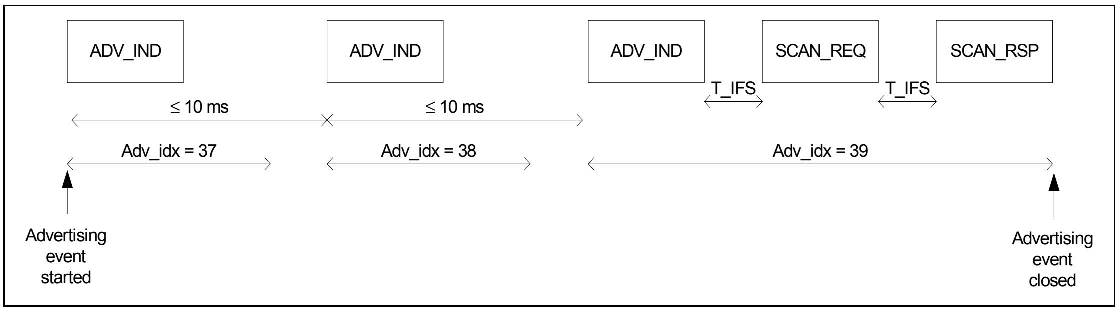


Figure 4.9: Connectable and scannable undirected advertising event with SCAN_REQ and SCAN_RSP PDUs at the end of an advertising event

Figure 4.10 illustrates an advertising event during which a CONNECT_IND PDU is received on the second primary advertising channel index.

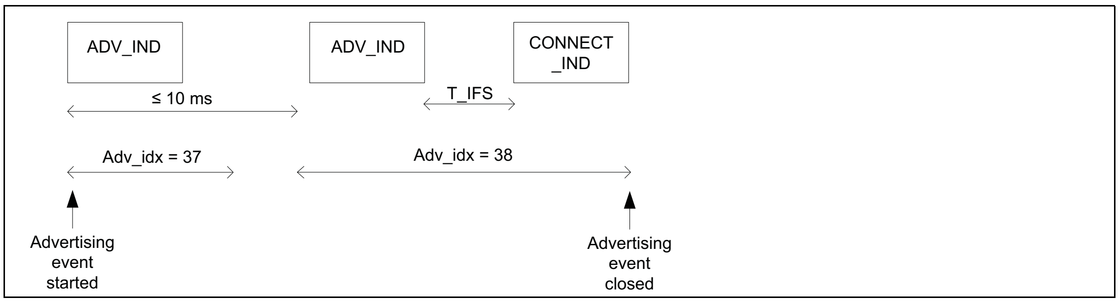


Figure 4.10: Connectable and scannable undirected advertising event when a CONNECT_IND PDU is received

If Link Layer Privacy has been enabled then the requirements in Section 6.2.1 shall also be followed.

4.4.2.4 Connectable Directed Event Type

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

The connectable directed advertising event type allows an initiator to respond so that both the advertiser and initiator will enter the Connection State.

The connectable directed advertising event type may use either the ADV_DIRECT_IND PDU (see Sections 4.4.2.4.1 to 4.4.2.4.3) or the ADV_EXT_IND PDU (see Section 4.4.2.4.4). A single connectable directed advertising event shall only use one of these two PDU types.

If Link Layer Privacy has been enabled then the requirements in Section 6.2.2 shall also be followed.

4.4.2.4.1 Connectable Directed Event Type using ADV_DIRECT_IND

This procedure shall not be used when the connectable directed event type is used on the LE Coded PHY.

The connectable directed advertising event type using ADV_DIRECT_IND allows an initiator to respond with a connect request on the primary advertising channel to establish a Link Layer connection.

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_IND PDU to the advertiser.

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

If the advertiser receives a CONNECT_IND 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 primary advertising channel index to send another ADV_DIRECT_IND PDU, or close the advertising event.

Connectable directed advertising may be either used in a low duty cycle or high duty cycle mode; these are described in the next two sections. Low duty cycle connectable directed advertising is designed for cases where reconnection with a specific device is required, but time is not of the essence or it is not known if the central device is in range or not. High duty cycle connectable directed advertising is designed for cases in which fast Link Layer connection setup is essential (for example, a reconnection). Note that high duty cycle connectable directed advertising is a power and bandwidth intensive advertising scheme that should only be used when fast connection setup is required.

4.4.2.4.2 Low Duty Cycle Connectable Directed Advertising

In low duty cycle connectable directed advertising, the time between the start of two consecutive ADV_DIRECT_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 an advertising event using all primary advertising channel indices and in which no CONNECT_IND PDUs are received is shown in Figure 4.11.

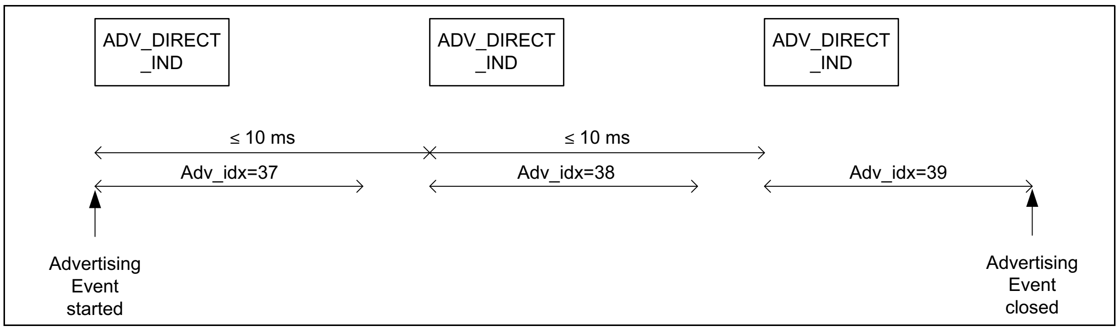


Figure 4.11: Low duty cycle connectable directed advertising event with only advertising PDUs

Figure 4.12 illustrates an advertising event using ADV_DIRECT_IND advertising PDUs during which a CONNECT_IND PDU is received on the second primary advertising channel index.

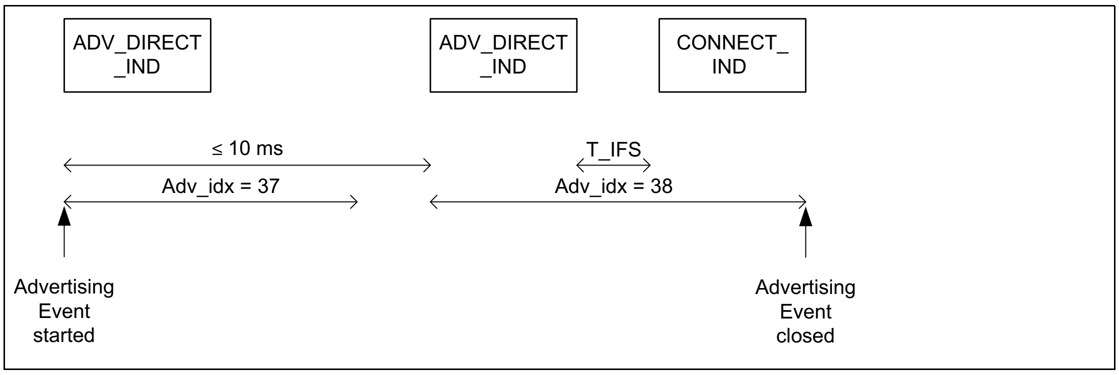


Figure 4.12: Low duty cycle connectable directed advertising event during which a CONNECT_IND PDU is received

4.4.2.4.3 High Duty Cycle Connectable Directed Advertising

In high duty cycle connectable directed advertising mode, 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_IND PDUs is shown in Figure 4.13 for the case in which all the primary advertising channels are used.

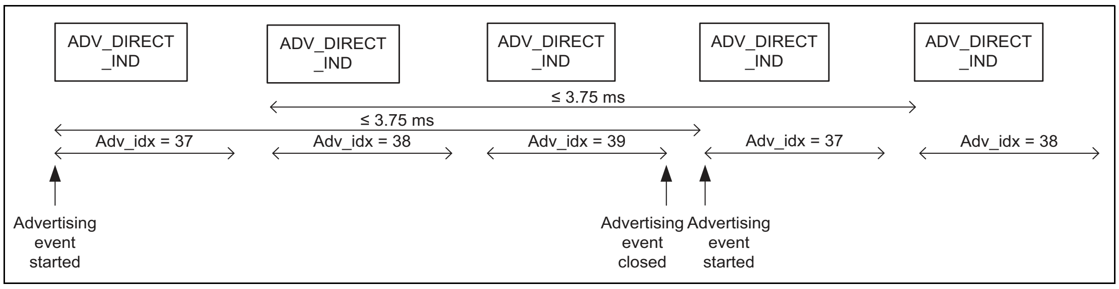


Figure 4.13: High duty cycle connectable directed advertising event with only advertising PDUs

4.4.2.4.4 Connectable Directed Event Type using ADV_EXT_IND

This procedure shall be used when the connectable directed event type is used on the LE Coded PHY.

The connectable directed advertising event type using ADV_EXT_IND allows an initiator to respond with a connect request on the secondary advertising channel to establish a Link Layer connection.

In the ADV_EXT_IND PDU, the AdvMode field shall be set to connectable, the ADI field shall be present, and the PDU shall not contain the AdvA and TargetA fields. The ADV_EXT_IND PDU’s AuxPtr field shall point to an AUX_ADV_IND PDU with the AdvMode field set to connectable; the AdvA, TargetA, and ADI fields shall all be present. The ADI fields in the ADV_EXT_IND PDU and the AUX_ADV_IND PDU shall be the same.

After every AUX_ADV_IND PDU related to this event it sends, the advertiser shall listen for AUX_CONNECT_REQ PDUs on the same secondary advertising channel index. Any AUX_SCAN_REQ PDUs received shall be ignored.

If the advertiser receives an AUX_CONNECT_REQ PDU that contains its device address and the initiator’s device address was contained in the AUX_ADV_IND PDU, it shall reply with an AUX_CONNECT_RSP PDU that contains those addresses on the same secondary advertising channel index. After the AUX_CONNECT_RSP PDU is sent 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. Any AUX_SCAN_REQ PDUs received on the secondary advertising channel shall be ignored.

The time between the start of two consecutive connectable directed ADV_EXT_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.

The channel index on the secondary channel, SAdv_idx, is contained in the AuxPtr field of the ADV_EXT_IND PDU.

Figure 4.14 shows an advertising event in which no AUX_CONNECT_REQ PDU is received.

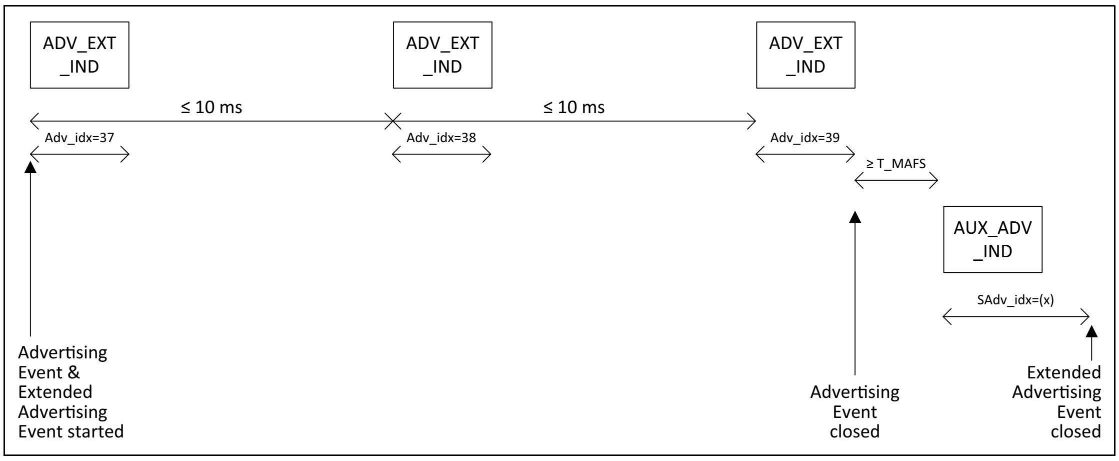


Figure 4.14: Connectable directed advertising event using the ADV_EXT_IND PDUs and AUX_ADV_IND PDU containing advertising data

Figure 4.15 illustrates an advertising event using connectable directed ADV_EXT_IND advertising PDUs during which an AUX_CONNECT_REQ PDU is received on the second secondary advertising channel index.

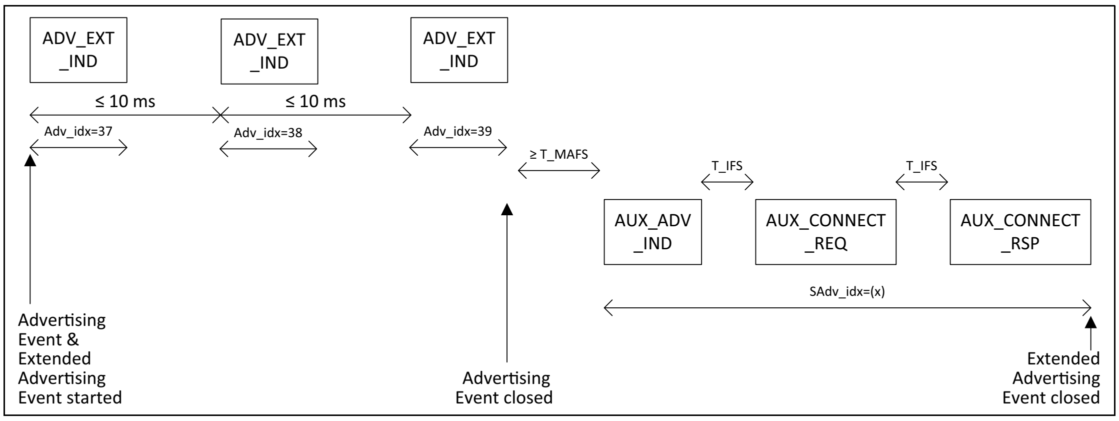


Figure 4.15: Connectable directed advertising event using ADV_EXT_IND PDUs and AUX_ADV_IND PDUs containing advertising data with an AUX_CONNECT_REQ PDU

4.4.2.5 Scannable Undirected Event Type

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

A scannable undirected advertising event shall use either ADV_SCAN_IND or scannable undirected ADV_EXT_IND PDUs but not both.

If Link Layer Privacy has been enabled then the requirements in Section 6.2.3 shall also be followed.

4.4.2.5.1 Scannable Undirected Event Type using ADV_SCAN_IND

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 primary advertising channel index for requests from scanners. Any CONNECT_IND PDUs received shall be ignored.

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 primary 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 10 ms. 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 4.16 for the case in which all the primary advertising channels are used.

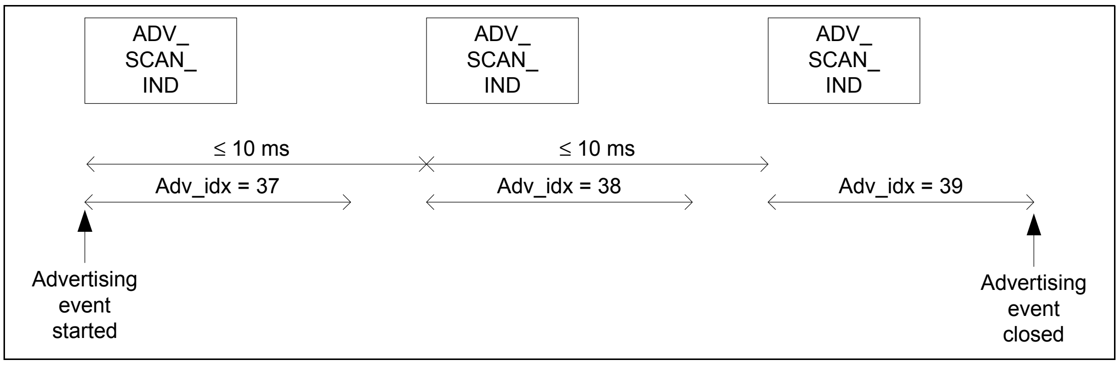


Figure 4.16: Scannable undirected advertising event with only advertising PDUs

Two example advertising events during which a SCAN_REQ PDU is received and a SCAN_RSP PDU is sent are shown in Figure 4.17 and in Figure 4.18 for the case in which all the primary advertising channels are used.

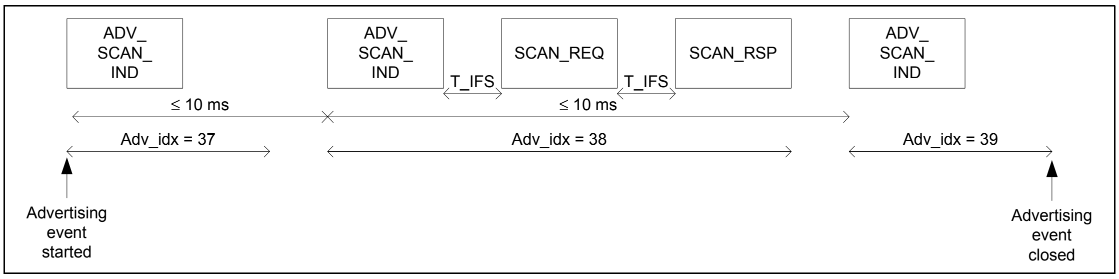


Figure 4.17: Scannable undirected advertising event with SCAN_REQ and SCAN_RSP PDUs in the middle of an advertising event

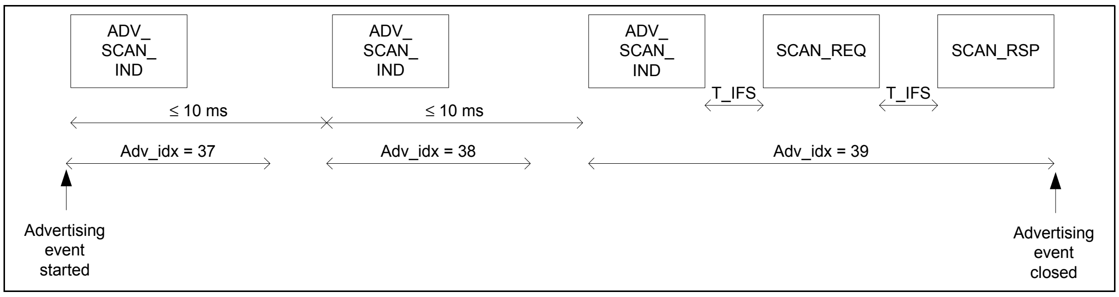


Figure 4.18: Scannable undirected advertising event with SCAN_REQ and SCAN_RSP PDUs at the end of an advertising event

4.4.2.5.2 Scannable Undirected Event Type using ADV_EXT_IND

The scannable undirected event type using the ADV_EXT_IND PDU allows any scanner to respond with a scan request to receive scan response data on the secondary advertising channel.

In the ADV_EXT_IND PDU the AdvMode field shall be set to scannable, the ADI field shall be present, and the PDU shall not contain the AdvA field. The ADV_EXT_IND PDU's AuxPtr field shall point to an AUX_ADV_IND PDU with the AdvMode field set to scannable; the AdvA and ADI fields shall be present. The ADI fields in the ADV_EXT_IND PDU and the AUX_ADV_IND PDU shall be the same. A scanner may send a scan request using the AUX_SCAN_REQ PDU on the same secondary advertising channel index as the received AUX_ADV_IND PDU pointed to by the ADV_EXT_IND PDU.

After every AUX_ADV_IND PDU sent by the advertiser, the advertiser shall listen for AUX_SCAN_REQ PDUs on the same secondary advertising channel index from scanners. Any AUX_CONNECT_REQ PDUs received shall be ignored.

If the advertiser receives an AUX_SCAN_REQ PDU that contains its device address from a scanner allowed by the advertising filter policy, it shall reply with an AUX_SCAN_RSP PDU on the same secondary advertising channel index prior to the start of the next advertising event. After the AUX_SCAN_RSP PDU is sent, or if the advertising filter policy prohibits processing the AUX_SCAN_REQ PDU, the advertising event shall be closed. Any AUX_CONNECT_REQ PDUs on the secondary advertising channel shall be ignored.

The time between the beginning of two consecutive ADV_EXT_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.

Figure 4.19 shows an advertising event in which no AUX_SCAN_REQ PDU is received.

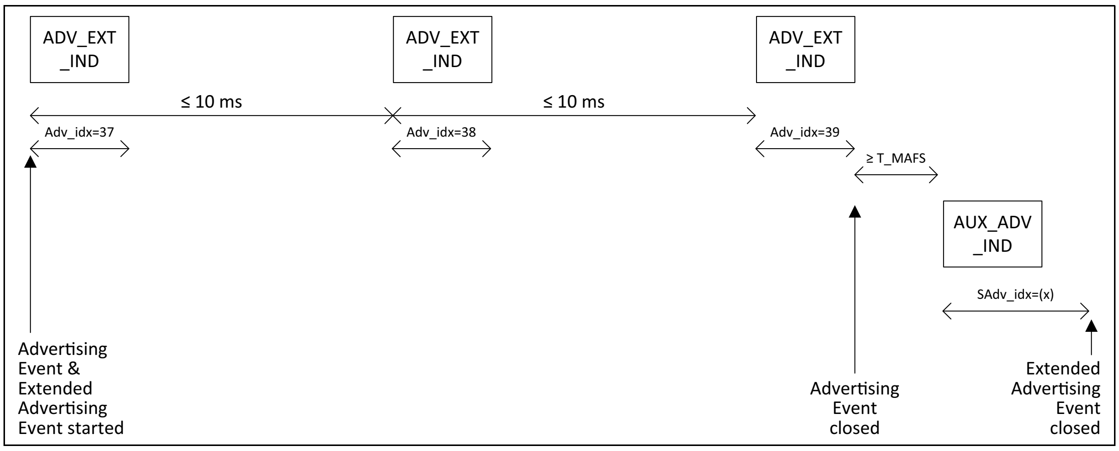


Figure 4.19: Scannable undirected advertising event using the ADV_EXT_IND PDUs and AUX_ADV_IND PDU containing advertising data

An example advertising event during which an AUX_SCAN_REQ PDU is received and an AUX_SCAN_RSP PDU is sent on the secondary advertising channel is shown in Figure 4.20.

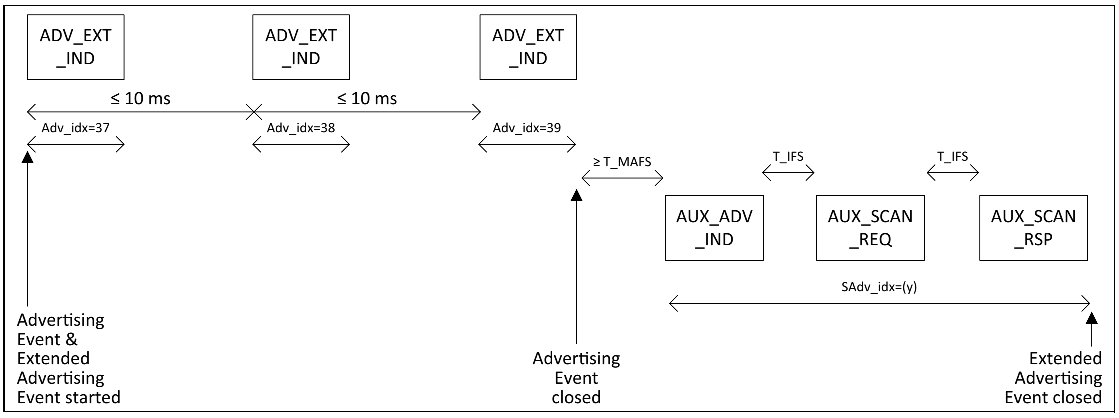


Figure 4.20: Scannable undirected advertising event with ADV_EXT_IND and AUX_ADV_IND PDUs with AUX_SCAN_REQ and AUX_SCAN_RSP PDUs on the secondary advertising channel

4.4.2.6 Non-Connectable and Non-Scannable Undirected Event Type

When the non-connectable and non-scannable undirected advertising event type is used, non-connectable and non-scannable undirected advertising indications (ADV_NONCONN_IND or non-connectable and non-scannable undirected ADV_EXT_IND PDUs) are sent by the Link Layer. The non-connectable and non-scannable undirected advertising event shall use either ADV_NONCONN_IND or non-connectable and non-scannable undirected ADV_EXT_IND PDUs but not both. The ADV_NONCONN_IND PDU shall not be used on the LE Coded PHY.

The non-connectable and non-scannable undirected event type allows a scanner to receive information from the advertiser. This information is contained either in the ADV_NONCONN_IND PDU or in an AUX_ADV_IND PDU pointed to by the AuxPtr field of the ADV_EXT_IND PDU.

The advertiser shall either move to the next used primary advertising channel index or close the advertising event after each ADV_NONCONN_IND or ADV_EXT_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 or ADV_EXT_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.

If an ADV_EXT_IND PDU is used, the AdvMode shall be set to non-connectable and non-scannable.

If an ADV_EXT_IND PDU is used, the Controller may use an auxiliary packet. It shall use an auxiliary packet if the advertisement includes ACAD or AdvData.

If an ADV_EXT_IND PDU is used without an auxiliary packet, it shall not contain an AuxPtr field and shall contain an AdvA field.

If an ADV_EXT_IND PDU is used with an auxiliary packet, it shall contain an AuxPtr field and an ADI field. Either the ADV_EXT_IND PDU or the AUX_ADV_IND that it points to, but not both, may contain the AdvA field (the AdvA field may be omitted entirely, in which case the advertising is anonymous). In the AUX_ADV_IND PDU the AdvMode shall be set to non-connectable and non-scannable and the ADI field shall be present. The ADI fields in the ADV_EXT_IND PDU and the AUX_ADV_IND PDU shall be the same. On the LE Coded PHY, the ADV_EXT_IND PDU shall not contain an AdvA field.

Note: The Controller can decide which PDU contains the AdvA field and should make this choice based on overall efficient use of the medium.

The TargetA field shall not be present in any PDU.

An illustration of a non-connectable and non-scannable undirected advertising event is shown in Figure 4.21 for the case in which all the primary advertising channels are used.

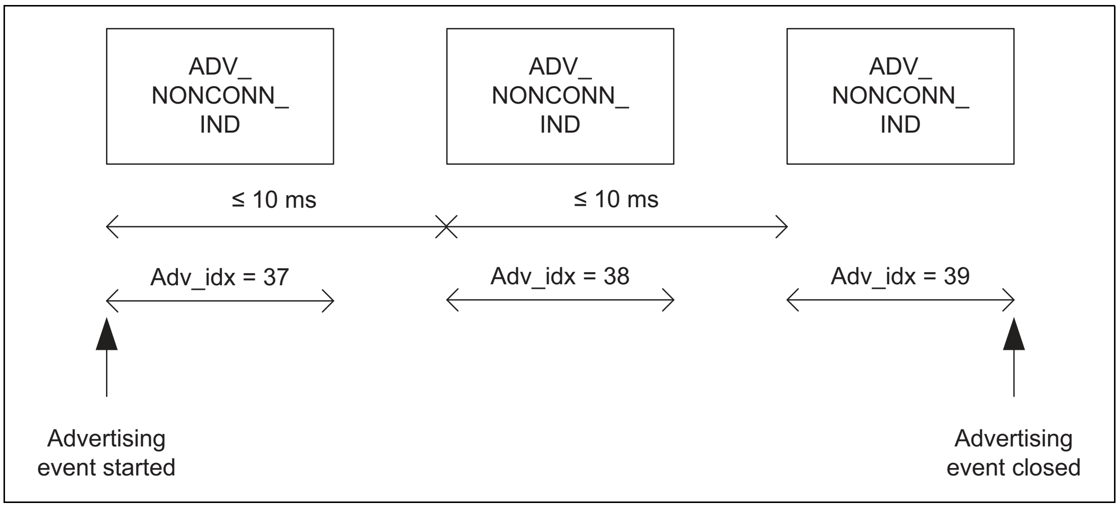


Figure 4.21: Non-connectable and non-scannable undirected advertising event using ADV_NONCONN_IND PDUs

Figure 4.22 shows an example of a non-connectable and non-scannable undirected ADV_EXT_IND PDU.

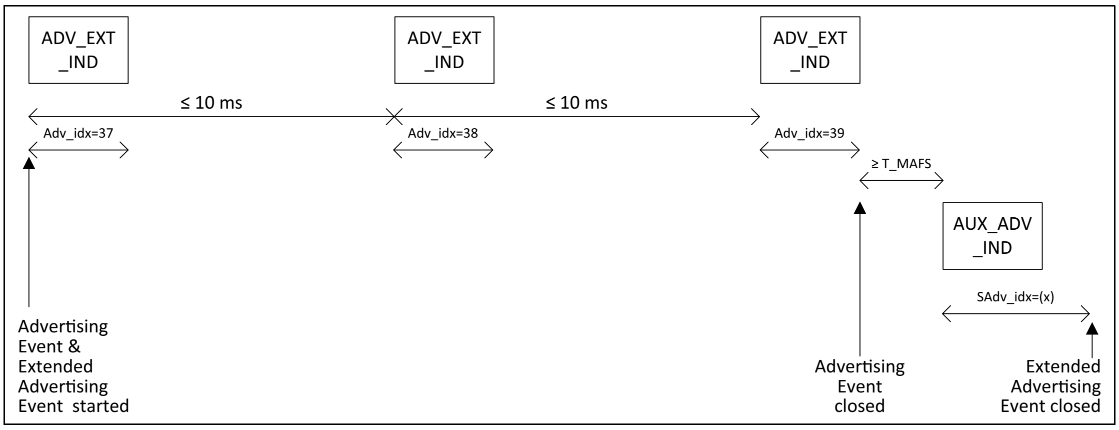


Figure 4.22: Non-connectable and non-scannable undirected advertising event using the ADV_EXT_IND PDU

Figure 4.23 shows an example of a non-connectable and non-scannable undirected ADV_EXT_IND PDU where the Host advertising data is fragmented using the AUX_CHAIN_IND PDU.

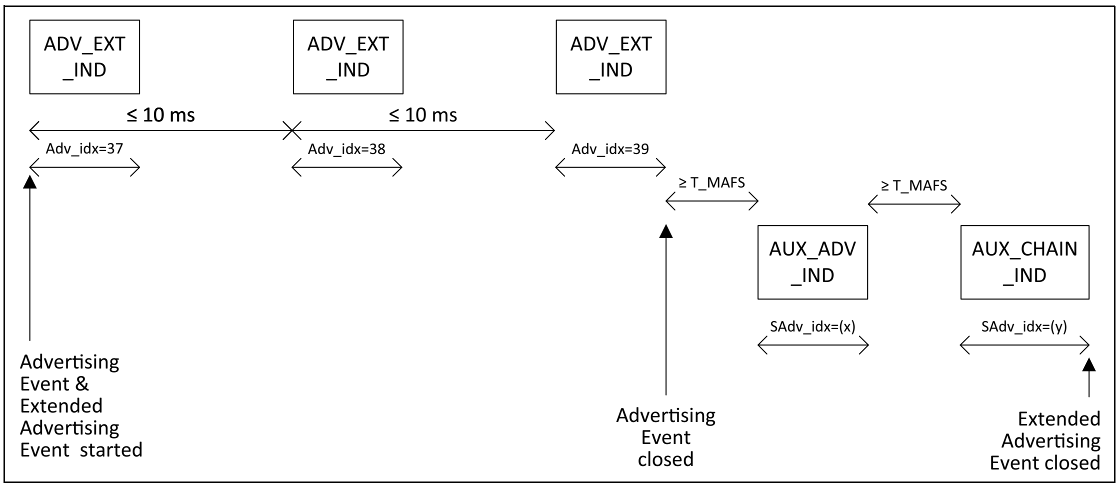


Figure 4.23: Non-connectable and non-scannable undirected advertising event using the ADV_EXT_IND PDU with fragmented Host advertising data

If Link Layer Privacy has been enabled then the requirements in Section 6.2.3 shall also be followed.

4.4.2.7 Connectable Undirected Event Type

The connectable undirected advertising event type using the ADV_EXT_IND PDU allows an initiator to respond with a connect request to establish a link layer connection on the secondary advertising channel.

In the ADV_EXT_IND PDU, the AdvMode field shall be set to connectable, the ADI field shall be present, and the PDU shall not contain the AdvA and TargetA fields. The AuxPtr field shall point to an AUX_ADV_IND PDU with the AdvMode field set to connectable and the AdvA and ADI fields present. The ADI fields in the ADV_EXT_IND PDU and the AUX_ADV_IND PDU shall be the same.

An initiator may send a connect request using the AUX_CONNECT_REQ PDU on the same secondary advertising channel as the AUX_ADV_IND PDU to request the Link Layer to enter the Connection State.

After every AUX_ADV_IND PDU sent related to this event by the advertiser, the advertiser shall listen for AUX_CONNECT_REQ PDUs on the same secondary advertising channel index. Any AUX_SCAN_REQ PDUs received shall be ignored.

If the advertiser receives an AUX_CONNECT_REQ PDU that contains its device address from an initiator allowed by the advertising filter policy it shall reply with an AUX_CONNECT_RSP PDU on the same secondary advertising channel index. After the AUX_CONNECT_RSP PDU is sent 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.

The time between the beginning of two consecutive ADV_EXT_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.

Figure 4.24 shows an advertising event in which no AUX_CONNECT_REQ PDU is received.

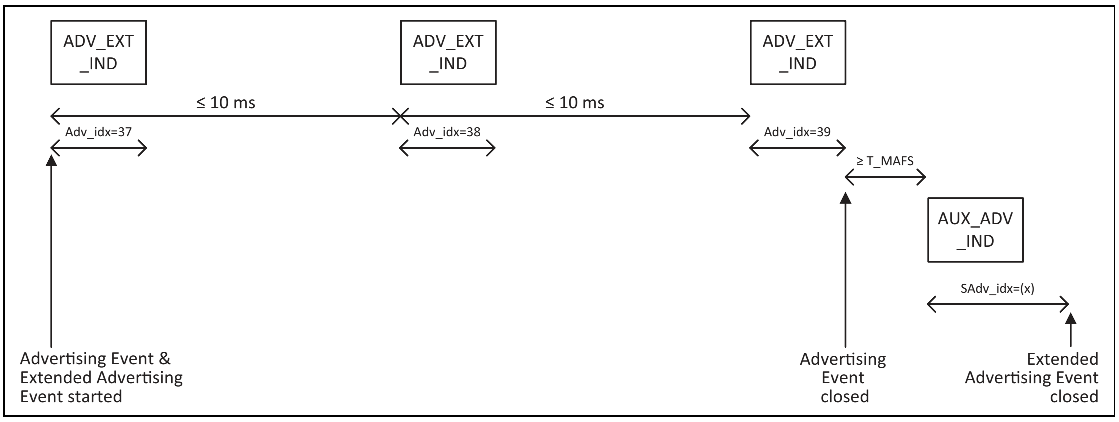


Figure 4.24: Connectable undirected advertising event using the ADV_EXT_IND PDUs and AUX_ADV_IND PDU containing advertising data

Figure 4.25 illustrates an advertising event during which an AUX_CONNECT_REQ PDU is received and an AUX_CONNECT_RSP PDU is sent on the secondary advertising channel index.

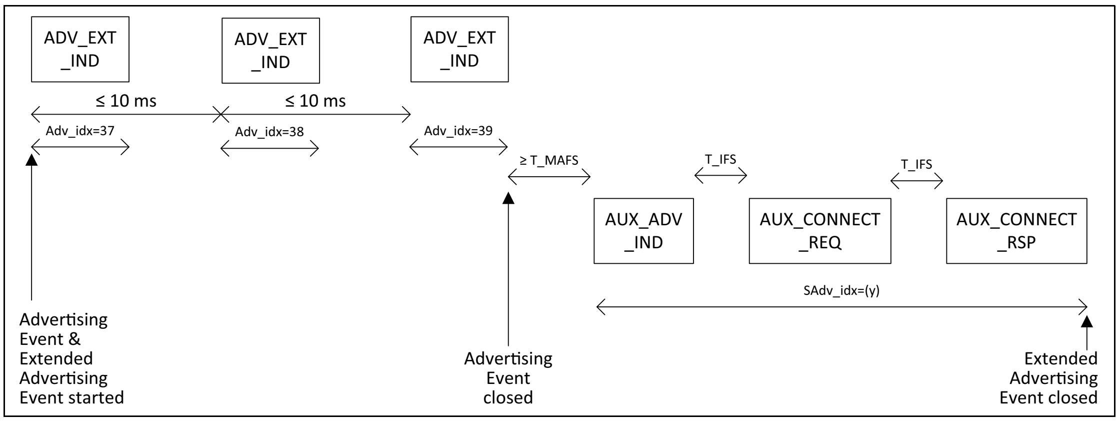


Figure 4.25: Connectable undirected advertising using ADV_EXT_IND PDUs when an AUX_CONNECT_REQ PDU is received

If Link Layer Privacy has been enabled then the requirements in Section 6.2.4 shall also be followed.

4.4.2.8 Scannable Directed Event Type

The scannable directed advertising event type using the ADV_EXT_IND PDU allows a specific scanner to respond with a scan request to receive scan response data on the secondary advertising channel.

In the ADV_EXT_IND PDU the AdvMode field shall be set to scannable, the ADI field shall be present, and the PDU shall not contain the AdvA and TargetA fields. The ADV_EXT_IND PDU's AuxPtr field shall point to an AUX_ADV_IND PDU with the AdvA, TargetA, and ADI fields all present. The ADI fields in the ADV_EXT_IND PDU and the AUX_ADV_IND PDU shall be the same.

After every AUX_ADV_IND PDU sent, the advertiser shall listen for an AUX_SCAN_REQ PDU on the same secondary advertising channel index from the targeted scanner.

If the advertiser receives an AUX_SCAN_REQ PDU that contains its device address and the scanner's device address is contained in the AUX_ADV_IND PDU, it shall reply with an AUX_SCAN_RSP PDU on the same secondary advertising channel index prior to the next advertising event. The AUX_SCAN_RSP PDU shall contain the scan response data. AUX_SCAN_REQ PDUs from any other scanner or any AUX_CONNECT_REQ PDUs received shall be ignored.

The time between the beginning of two consecutive ADV_EXT_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.

See Section 4.4.2.5.2 for a description on how the AUX_SCAN_REQ packet is used in conjunction with the AUX_SCAN_RSP packets on the secondary advertising channel.

If Link Layer Privacy has been enabled then the requirements in Section 6.2.5 shall also be followed.

4.4.2.9 Non-Connectable and Non-Scannable Directed Event Type

The non-connectable and non-scannable directed advertising event type using the ADV_EXT_IND PDU allows an advertiser to send non-connectable and non-scannable directed ADV_EXT_IND PDUs on the primary advertising channel with any advertising data sent on the secondary advertising channel targeted for a specific scanner.

The AdvMode field in the ADV_EXT_IND PDU shall be set to non-connectable and non-scannable.

The Controller shall use an auxiliary packet if the advertisement includes ACAD or AdvData. Otherwise the Controller may use an auxiliary packet.

If an auxiliary packet is not used, the ADV_EXT_IND PDU shall not contain an AuxPtr field and shall contain the AdvA and TargetA fields.

If an auxiliary packet is used, the ADV_EXT_IND PDU shall contain an AuxPtr field and an ADI field. Either the ADV_EXT_IND PDU or the AUX_ADV_IND PDU it points to, but not both, may contain the AdvA field (the AdvA field may be omitted entirely, in which case the advertising is anonymous). The TargetA field shall be present in either PDU but not both. In the AUX_ADV_IND PDU the AdvMode shall be set to non-connectable and non-scannable and the ADI field shall be present. The ADI fields in the ADV_EXT_IND PDU and the AUX_ADV_IND PDU shall be the same. On the LE Coded PHY, the ADV_EXT_IND PDU shall not contain an AdvA or TargetA field.

Note: The Host cannot specify which PDU contains the AdvA or TargetA field; the Controller should make this choice based on overall efficient use of the medium.

The Link Layer does not listen, and therefore cannot receive any requests from scanners or initiators.

If Link Layer Privacy has been enabled then the requirements in Section 6.2.5 shall also be followed.

4.4.2.10 Advertising Data Sets

The advertiser's Host may instruct the Link Layer to interleave advertising events. Advertising data belonging together is called an advertising data set. The Link Layer may support multiple advertising data sets, with each set having different advertising parameters such as advertising PDU type, advertising interval, and PHY.

When advertising with the ADV_EXT_IND, AUX_ADV_IND, or AUX_SYNC_IND PDUs, the advertising data set is identified by the Advertising SID subfield of the ADI field. The Link Layer shall set the Advertising SID subfield as directed by the Host.

The scanner may filter advertisements based on the Advertising SID.

The advertising events for each Advertising Data Set are considered a separate instance of the Advertising State and each have their own Advertising Interval (see Section 4.4.2.2.1).

Figure 4.26 illustrates an example of advertising using multiple advertising sets.

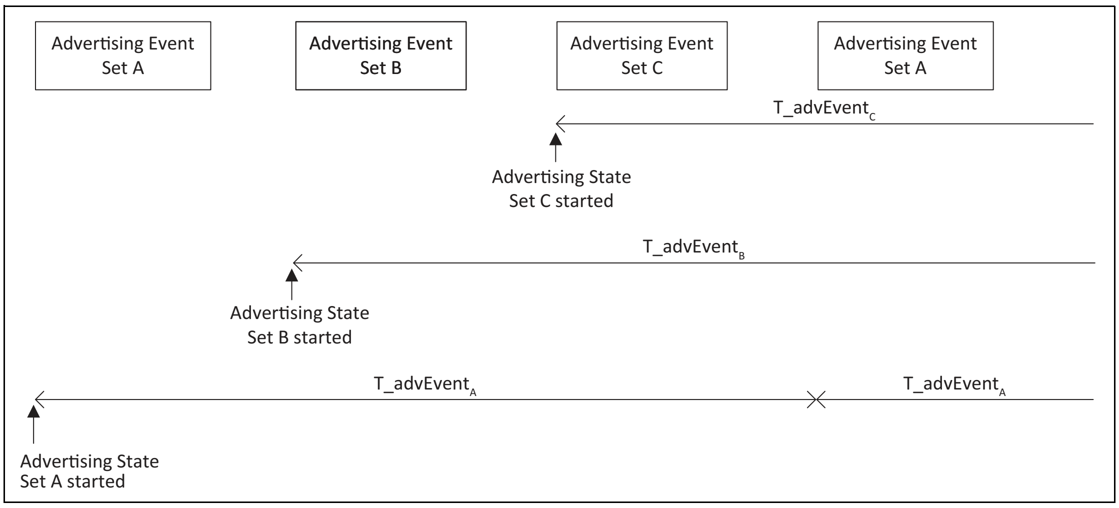


Figure 4.26: Multiple advertising data sets example

On reset, all advertising sets are destroyed.

When an advertising set is created that includes advertising data, the Controller shall guarantee that the set can contain at least 31 octets of advertising data. The guarantee shall no longer apply after the Host first specifies advertising data for that set or creates another advertising set.

When an advertising set is created that is scannable, the Controller shall guarantee that the set can contain at least 31 octets of scan response data. The guarantee shall no longer apply if the set is made non-scannable or after the Host first specifies scan response data for that set or creates another advertising set.

4.4.2.11 Using AdvDataInfo (ADI)

The AdvDataInfo (ADI) field is used to identify advertising sets and duplicate AdvData in either the AUX_ADV_IND or AUX_SCAN_RSP PDUs. For scannable advertising events using the ADV_EXT_IND PDU, AdvData is not permitted in the AUX_ADV_IND PDU so the ADI only refers to the AdvData contained in the AUX_SCAN_RSP PDU.

The Advertising DID for a given advertising set shall be initialized with a randomly chosen value. Whenever the Host provides new advertising data or scan response data for a given advertising set (whether it is the same as the previous data or not), the Advertising DID shall be updated. The new value shall be a randomly chosen value that is not the same as the previously used value.

Note: Choosing Advertising DID field values randomly reduces the possibility of PDUs from different advertisers containing the same ADI field value.

4.4.2.12 Periodic Advertising

When advertising data is required to be sent regularly at a fixed interval, periodic advertising is used. Periodic advertising consists of advertisements sent at a fixed interval with the advertisement data changing from time to time.

When periodic advertising takes place, the advertiser shall send AUX_SYNC_IND PDUs at regular intervals (the periodic advertising interval - see Section 4.4.2.2.3), which are described in the SyncInfo field of AUX_ADV_IND PDUs. The Periodic Advertising is identified by the AdvDataInfo field of ADV_EXT_IND PDUs that point to AUX_ADV_IND PDUs containing the SyncInfo field. The AUX_SYNC_IND PDUs and the PDUs that point to them shall always be sent on the same PHY. The PHY used shall not change while the periodic advertising is enabled. Advertising pointing to a periodic advertisement shall not be anonymous. Each time that periodic advertising is enabled, the Controller shall transmit at least one AUX_IND_PDU pointing to the first AUX_SYNC_IND PDU of that periodic advertising; after this, there is no requirement whether or when to transmit advertising PDUs pointing to the periodic advertisements.

The Host may send periodic advertising data to the Link Layer. This advertising data is placed by the Link Layer in the periodic AUX_SYNC_IND PDUs and their subordinate sets. The Link Layer shall repeat the last advertising data sent by the Host until it receives new advertising data. The AUX_SYNC_IND PDUs are continuously sent out until the Host directs the Link Layer to terminate the periodic advertising.

When an advertising set is first configured for periodic advertising, the Controller shall guarantee that the set can contain at least 31 octets of advertising data or else shall not allow periodic advertising on that advertising set. The guarantee shall no longer apply after the Host first specifies periodic advertising data for that set or creates another advertising set.

Figure 4.27 illustrates an example of periodic advertising.

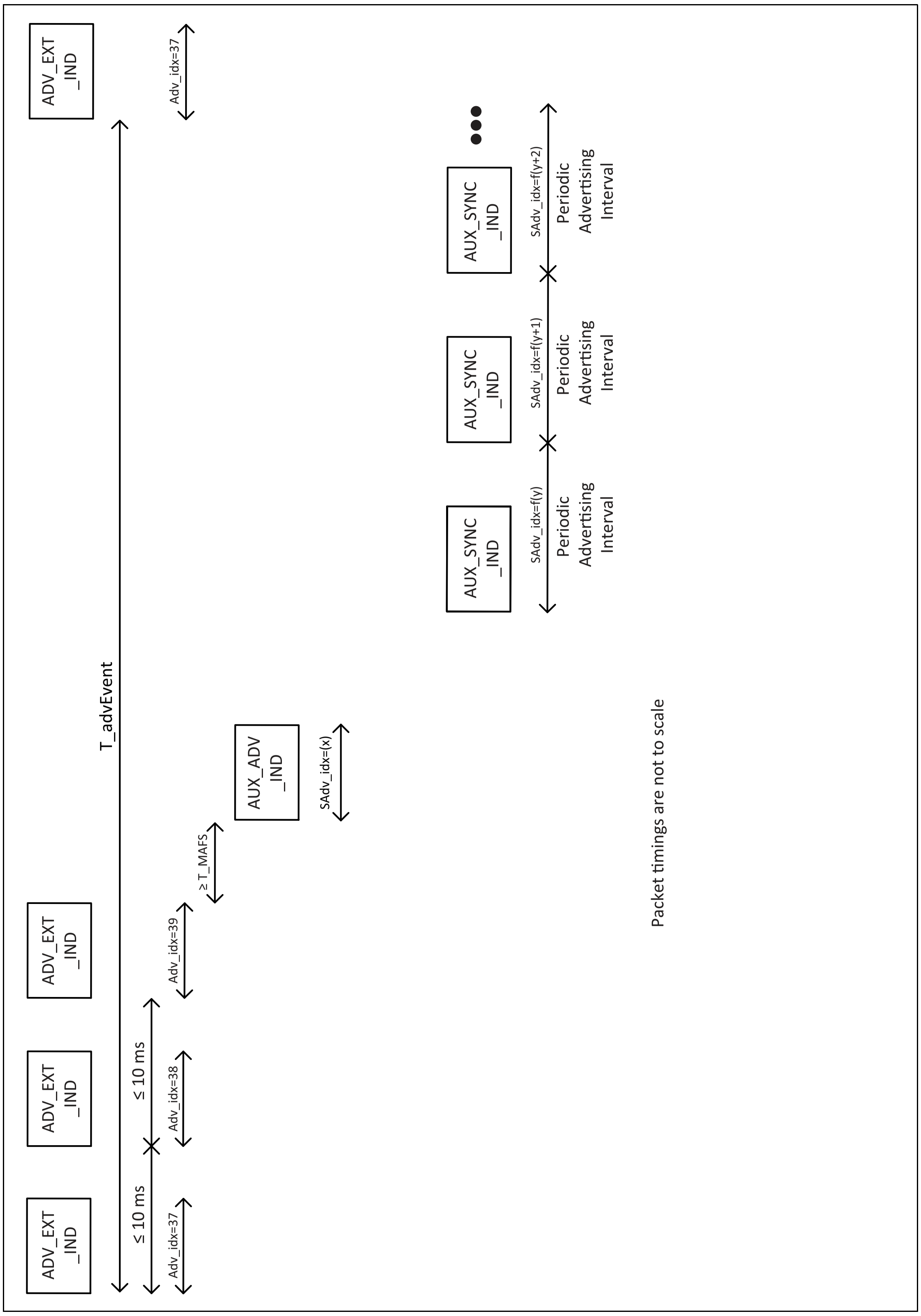


Figure 4.27: Example of periodic advertising

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 primary advertising channel. 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 a primary 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 primary advertising channel index. The Link Layer shall use all the primary advertising channel indices.

The *scanWindow* and *scanInterval* parameters shall be less than 40.96 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 or scanning PDU when scanning.

On receiving a PDU with the AuxPtr field present, the scanner should also listen for the auxiliary PDU it points to (provided that it supports the PHY specified in the AuxPtr field).

When a scanner receives ADV_EXT_IND PDUs that contain an AuxPtr field, it may either always listen for the auxiliary packet or may sometimes skip listening. In the latter case, the following requirements shall apply.

For each Advertising SID value received:

- The Controller shall keep a cache of one or more recent Advertising DID values used by each advertising device (for this purpose, all anonymous advertising is treated as being from a single device different to all real devices) and shall update them whenever a PDU containing an ADI field is received. The Controller may delete any cache entry at any time. The Controller should delete the cache entry relating to an ADV_EXT_IND PDU if it fails to receive that PDU's entire subordinate set.
- The Controller may only skip listening for the auxiliary packet if the cache has an entry specifying the Advertising DID value in the ADI field being used by a device; if the ADV_EXT_IND PDU contains an AdvA field, the entry shall be for that device. Otherwise, the Controller shall not skip listening for the auxiliary packet. The Controller should sometimes listen for the

AUX_ADV_IND PDU in case another advertiser has started using the same Advertising DID value.

If Link Layer Privacy has been enabled then the requirements in Section 6.3 shall also be followed.

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.

After entering the Scanning State, if the Link Layer receives a scannable PDU (i.e. an ADV_IND, ADV_SCAN_IND, or scannable AUX_ADV_IND PDU) from an advertiser allowed by the scanner filter policy, it shall respond with a scan request PDU and then listen for the scan response PDU. It shall continue to respond to the same advertiser until it has successfully received the scan response PDU. It may then either respond to or ignore subsequent scannable PDUs from the same advertiser. It should ignore them if either they are legacy PDUs or if the Advertising DID field has not changed since the last advertisement from the same advertiser with the same Advertising SID field; it should not ignore them otherwise.

The Link Layer shall only send a SCAN_REQ PDU to an advertiser from which an ADV_IND PDU or ADV_SCAN_IND PDU is received. The Link Layer shall only send an AUX_SCAN_REQ PDU to an advertiser from which a scannable AUX_ADV_IND is received. The Link Layer shall ignore a scannable ADV_EXT_IND or AUX_ADV_IND PDU if the TargetA field is present and it does not match the Link Layer's device address.

The scanner shall run a backoff procedure to minimize collisions of scan request PDUs from multiple scanners. An example of such a procedure is given in the following paragraphs.

The backoff procedure uses two parameters, *backoffCount* and *upperLimit,* to restrict the number of scan request PDUs sent when collisions occur on scan response PDUs. Upon entering the Scanning State, the *upperLimit* and *backoffCount* are set to one.

On every received ADV_IND, ADV_SCAN_IND, or scannable AUX_ADV_IND PDU that is allowed by the scanner filter policy and for which a scan request PDU is to be sent, the *backoffCount* is decremented by one until it reaches the value of zero. The scan request PDU is only sent when *backoffCount* becomes zero.

After sending a scan request PDU the Link Layer listens for a scan response PDU from that advertiser. If the scan response 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* is doubled until it reaches the value of 256. On every two consecutive successes, the *upperLimit* is halved until it reaches the value of one. After success or failure of receiving the scan response PDU, the Link Layer sets *backoffCount* to a new pseudo-random integer between one and *upperLimit* inclusive.

If a device uses a different backoff algorithm it shall share the medium responsibly.

Two illustrations of advertising events using all the advertising channel indices 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.

4.4.3.3 Advertising Data Sets

The ADV_EXT_IND PDU may contain an ADI field. When the ADI field is present, it can be used to identify advertisement data that belong to the same set. This is specified in the Advertising SID subfield in the ADI. The Advertising SID is set by the Host of the advertiser.

4.4.3.4 Periodic Advertisements

For an AUX_ADV_IND PDU where the SyncInfo field is present the scanner should, when instructed by the Host, listen for the AUX_SYNC_IND PDU specified in the SyncInfo field. The scanner should then continue to listen for AUX_SYNC_IND PDUs on the secondary advertising channel indices specified in Section 4.4.2.1.

Because of sleep clock accuracies (see Section 4.2.2), the scanner should perform the window widening specified in Section 4.5.7, with connection events replaced by packets containing AUX_SYNC_IND PDUs.

If the Sync Packet Offset of the SyncInfo is zero, the scanner should listen for a subsequent advertisement to be able to locate the periodic advertisement.

The Link Layer on the scanner shall report the advertising data received in the periodic advertisements to the Host.

A scanner shall not attempt to synchronize to a periodic advertising set that it is already synchronized to.

4.4.3.5 Advertising Reports

For each non-duplicate advertising or scan response PDU from an advertiser, the Link Layer shall send an advertising report to the Host. However, if the Controller receives an ADV_EXT_IND PDU with an AuxPtr field, it shall delay the report until after the corresponding AUX_ADV_IND PDU has been received and the report shall combine the information in the PDUs; if the Controller does not listen for or does not receive the AUX_ADV_IND PDU, no report shall be generated. The advertising report shall contain at least the advertiser's device address and advertising data or scan response data if present. The Host may request that duplicate advertising reports are filtered.

Where a received ADV_EXT_IND PDU contains an ADI field, a duplicate advertising report is an advertising report for the same device address where the previous report that contained an ADI value with the same Advertising SID also had the same Advertising DID. For this purpose, all anonymous advertising is treated as being from a single device different to all non-anonymous devices.

Where the ADV_EXT_IND PDU does not contain an ADI field or a legacy PDU was received, a duplicate advertising report is an advertising report for the same device address while the Link Layer stays in the Scanning State.

In either case the actual data may change; advertising data or scan response data is not considered significant when determining duplicate advertising reports.

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 primary advertising channel.

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

During initiating, the Link Layer listens on a primary 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 primary advertising channel index. The Link Layer shall use all the primary advertising channel indices.

The *scanWindow* and *scanInterval* parameters shall be less than or equal to 40.96 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.

Connection indications or requests in response to a connectable advertisement shall be sent on either the primary or secondary advertising channel depending on which advertising PDU contains an AdvA field. The following sub-sections describe the two procedures.

If Link Layer Privacy has been enabled then the requirements in Section 6.4 shall also be followed.

4.4.4.1 Connect Requests on the Primary Advertising Channel

This procedure shall not be used when establishing a connection on the LE Coded PHY.

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

After sending the CONNECT_IND 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.

4.4.4.2 Connect Requests on the Secondary Advertising Channel

This procedure shall be used when establishing a connection on the LE Coded PHY.

If a connectable ADV_EXT_IND PDU is received, the initiator shall listen for the connectable AUX_ADV_IND on the secondary advertising channel. If a connectable undirected AUX_ADV_IND PDU, or a connectable directed AUX_ADV_IND PDU containing the initiator's Link Layer device address, is received and is allowed by the initiator filter policy, the initiator shall send an AUX_CONNECT_REQ PDU to the advertiser; otherwise, it shall be ignored.

After sending the AUX_CONNECT_REQ PDU, the initiator shall wait for the advertiser to send an AUX_CONNECT_RSP PDU. Once an AUX_CONNECT_RSP PDU is received, 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. If the initiator does not receive an AUX_CONNECT_RSP PDU from the advertiser, it shall use the back-off algorithm described for SCAN_REQ in Section 4.4.3.2 before responding to the next connectable AUX_ADV_IND PDU.

4.5 CONNECTION STATE

The Link Layer enters the Connection State when an initiator sends a CONNECT_IND PDU on the primary advertising channel to an advertiser, an advertiser receives a CONNECT_IND PDU on the primary advertising channel from an initiator, an advertiser sends an AUX_CONNECT_RSP PDU on the secondary advertising channel to an initiator, or an initiator receives an AUX_CONNECT_RSP PDU on the secondary advertising channel from an advertiser.

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).

If the connection is first created using the CONNECT_IND PDU on the primary advertising channel, it shall use the LE 1M PHY in both directions. If the connection is first created on the secondary channel using the AUX_CONNECT_REQ and AUX_CONNECT_RSP PDUs, it shall use the same PHY in both directions as was used for the AUX_CONNECT_REQ and AUX_CONNECT RSP PDU. Either PHY may be changed subsequently using the PHY Update Procedure (Section 5.1.10). When the LE Coded PHY is in use, the coding of each packet is determined by the CI field as defined in Section 2.2.3 and may be different in each direction and in adjacent packets in a given direction.

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. There shall be only one connection, whether or not established, between two LE device addresses. An initiator shall not send a connection request to an advertiser it is already connected to.

If an advertiser receives a connection request from an initiator it is already connected to, it shall ignore that request.

If the initiator sent a CONNECT_IND PDU in response to an ADV_IND or ADV_DIRECT_IND PDU and either or both device’s PDU had the ChSel field set to 0, then Channel Selection Algorithm #1 (see Section 4.5.8.2) shall be used on the connection. Otherwise, Channel Selection Algorithm #2 (see Section 4.5.8.3) shall be used.

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 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 CONNECT_IND 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**2)) - 1). The *connSlaveLatency* parameter 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*), containing the value connEventCount, 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 is not 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* * 2.

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_IND and AUX_CONNECT_REQ PDUs include 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_IND and AUX_CONNECT_REQ PDUs include three parameters used to determine the transmit window. The transmit window starts at *transmitWindowDelay* + *transmitWindowOffset* after the end of the packet containing the CONNECT_IND PDU or AUX_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. The *transmitWindowOffset* and *transmitWindowSize* parameters 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 *transmitWindowDelay* + *transmitWindowOffset* and no later than *transmitWindowDelay* + *transmitWindowOffset* + *transmitWindowSize* after the end of the packet containing the CONNECT_IND PDU or AUX_CONNECT_REQ PDU.

The value of *transmitWindowDelay* shall be 1.25 ms when a CONNECT_IND PDU is used, 2.5 ms when an AUX_CONNECT_REQ PDU is used on an LE Uncoded PHY, and 3.75 ms when an AUX_CONNECT_REQ PDU is used on the LE Coded PHY.

4.5.4 Connection Setup – Master Role

After the initiator sends a CONNECT_IND PDU on the primary advertising channel or receives an AUX_CONNECT_RSP PDU on the secondary advertising channel, the Link Layer is in the 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 the master's perspective are shown in Figure 4.28 and in Figure 4.29.

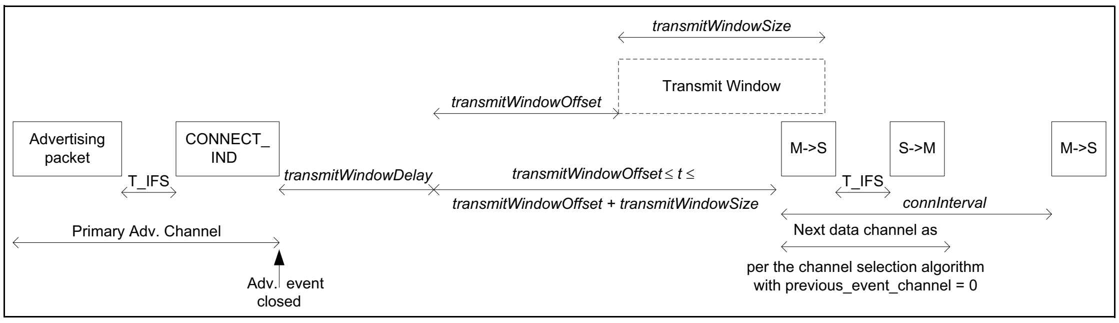


Figure 4.28: Master’s view of LL connection setup with CONNECT_IND

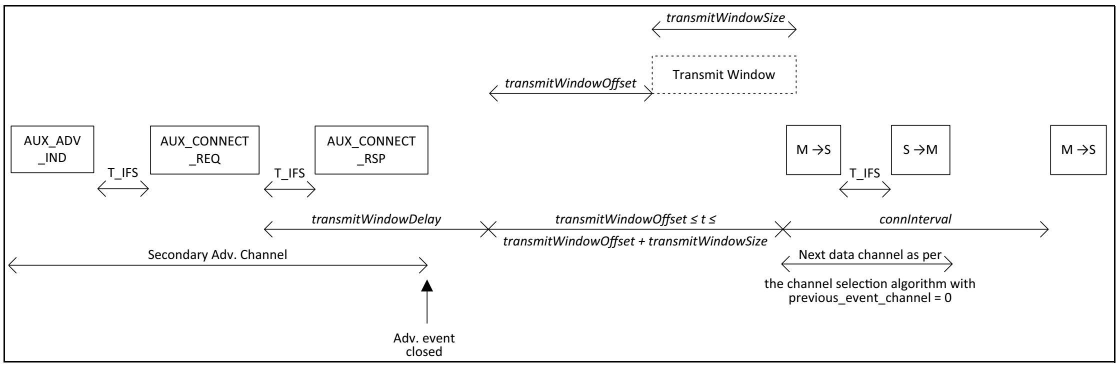


Figure 4.29: Master’s view of LL connection setup with AUX_CONNECT_REQ

4.5.5 Connection Setup – Slave Role

After the advertiser receives a CONNECT_IND PDU on the primary advertising channel or sends an AUX_CONNECT_RSP PDU on the secondary advertising channel, the Link Layer is in the 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.

Two examples of the procedure from the slave's perspective are shown in Figure 4.30 and in Figure 4.31. In these examples 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).

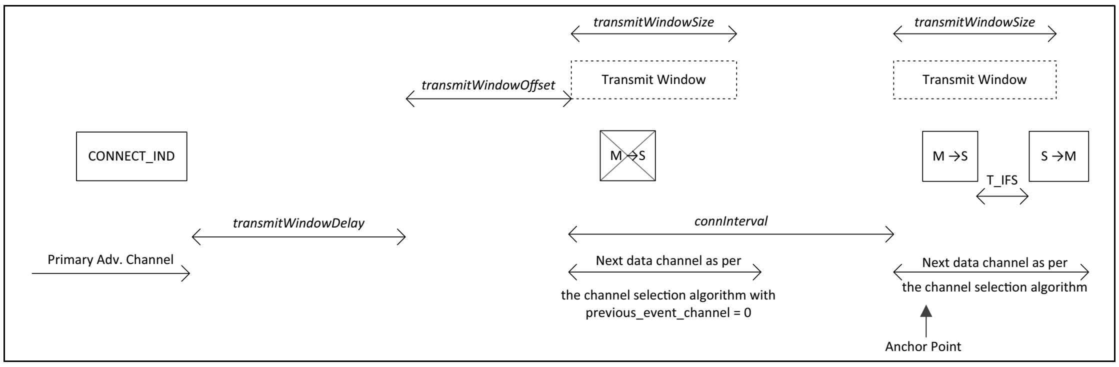


Figure 4.30: Slave closing LL connection setup in the second LL connection event with CONNECT_IND

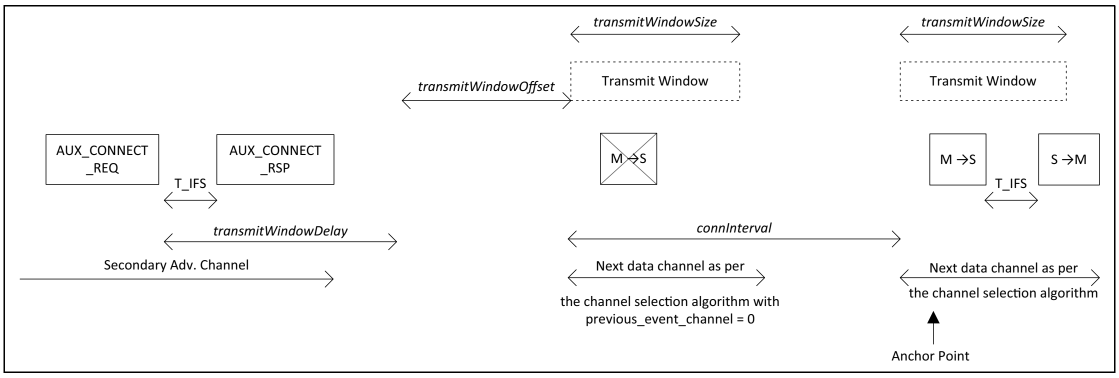


Figure 4.31: Slave closing LL connection setup in the second LL connection event with AUX_CONNECT_REQ

The slave shall be active in every connection event until it receives a packet from the master with the NESN set to one. From then on it may use slave latency 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. In the absence of more accurate information about the master's clock, the slave shall also use the master's sleep clock accuracy (*masterSCA*) from the CONNECT_IND 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:

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

If the slave has more accurate information about the master's clock, it may select a smaller value for *windowWidening*.

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 transmit window 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_IND 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 Algorithm #1

Channel Selection Algorithm #1 only supports channel selection for connection events.

Channel Selection Algorithm #1 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:

$$\mathit{unmappedChannel} = (\mathit{lastUnmappedChannel} + \mathit{hopIncrement}) \bmod 37$$

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, Channel Selection Algorithm #1 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 4.32.

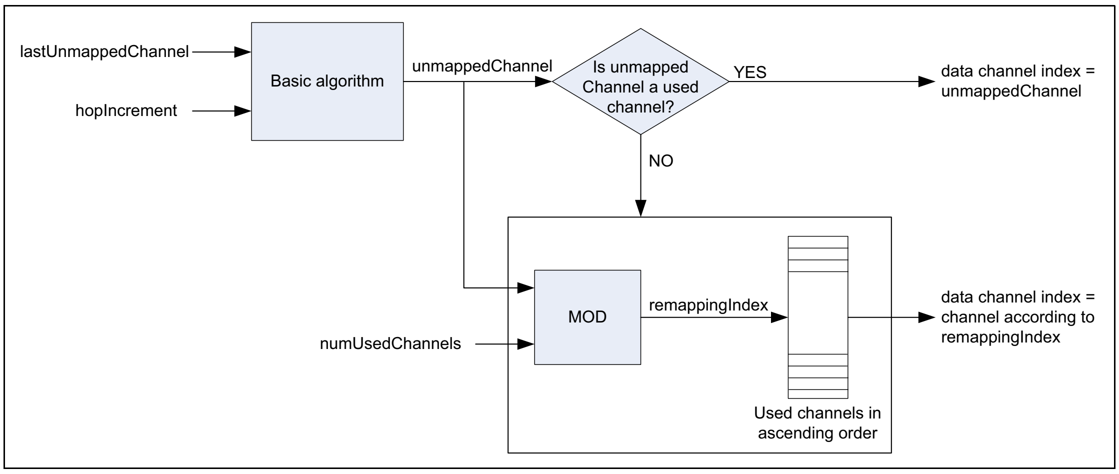

lastUnmappedChannel
hopIncrement
Basic algorithm
unmappedChannel
Is unmapped Channel a used channel?
YES
data channel index = unmappedChannel
NO
MOD
numUsedChannels
remappingIndex
data channel index = channel according to remappingIndex
Used channels in ascending order

Figure 4.32: Block diagram of data Channel Selection Algorithm #1

4.5.8.3 Channel Selection Algorithm #2

4.5.8.3.1 Overview

Channel Selection Algorithm #2 supports channel selection for connection events and periodic advertising packets.

At the start of an event, which can be a connection event or a periodic advertising packet, the algorithm described here generates an event channel index (which is a data channel index or secondary advertising channel index, as appropriate).

A block diagram of the overall algorithm is shown in Figure 4.33.

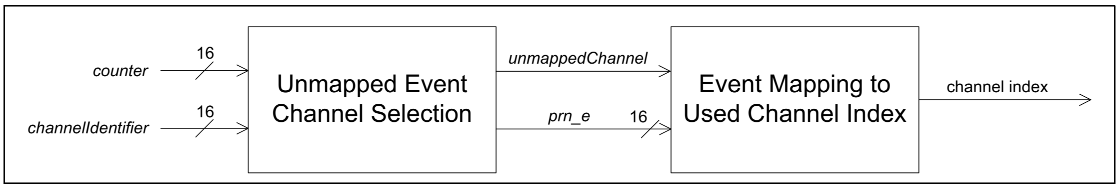

counter
16
channelIdentifier
16
Unmapped Event Channel Selection
unmappedChannel
prn_e
16
Event Mapping to Used Channel Index
channel index

Figure 4.33: General block diagram of Channel Selection Algorithm #2

4.5.8.3.2 Inputs and Basic Components

The algorithm makes use of the following inputs and basic components:

- The 6-bit input *N* is the number of channels classified as *Used* channels.
- The 16-bit input *channelIdentifier* is fixed for any given connection or periodic advertising; it is calculated from the Access Address by:

  channelIdentifier = (Access Address$_{31\text{-}16}$) XOR (Access Address$_{15\text{-}0}$)
- The 16-bit input *counter* changes for each event. For data connections it is the connection event counter *connEventCounter* defined in Section 4.5.1. For periodic advertising it is the event counter *paEventCounter* defined in Section 4.4.3.4.

The “XOR” operation always refers to a 16-bit bit-wise XOR.

The symbol $\lfloor\ \rfloor$ is used to represent the floor function (the greatest integer less than or equal to the argument).

The permutation operation consists of separately bit-reversing the lower 8 input bits and upper 8 input bits, as illustrated in Figure 4.34.

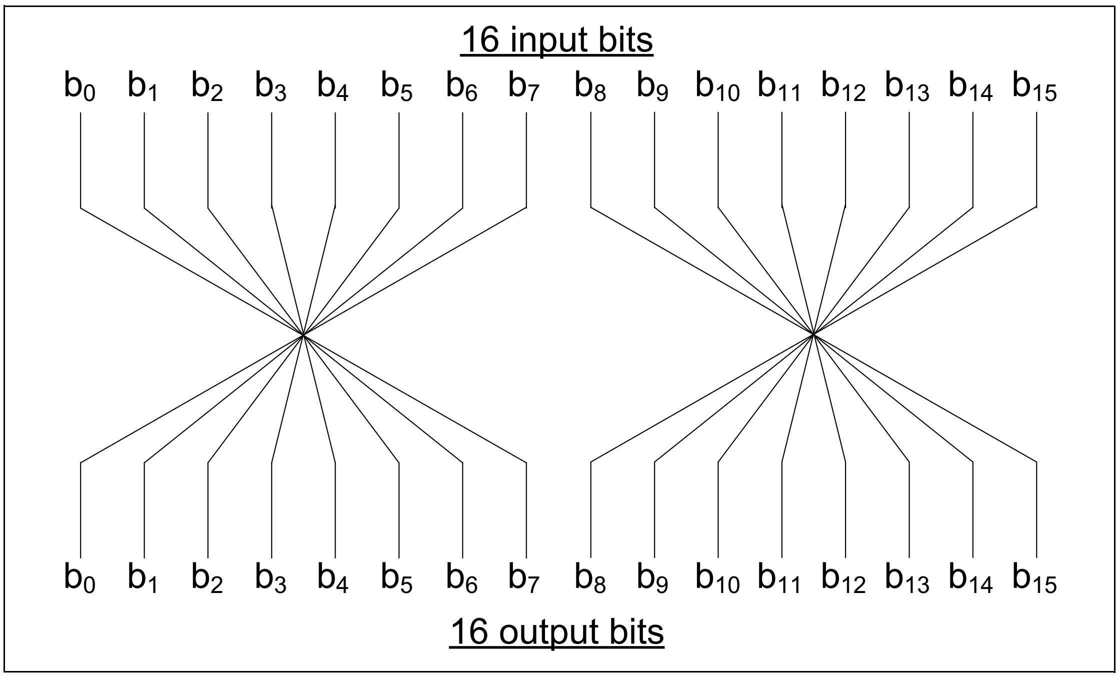

16 input bits
b0 b1 b2 b3 b4 b5 b6 b7 b8 b9 b10 b11 b12 b13 b14 b15
b0 b1 b2 b3 b4 b5 b6 b7 b8 b9 b10 b11 b12 b13 b14 b15
16 output bits

Figure 4.34: Permutation operation

The Multiply, Add, and Modulo (MAM) block performs a multiplication operation, an addition operation, and a modulo operation, as illustrated in Figure 4.35.

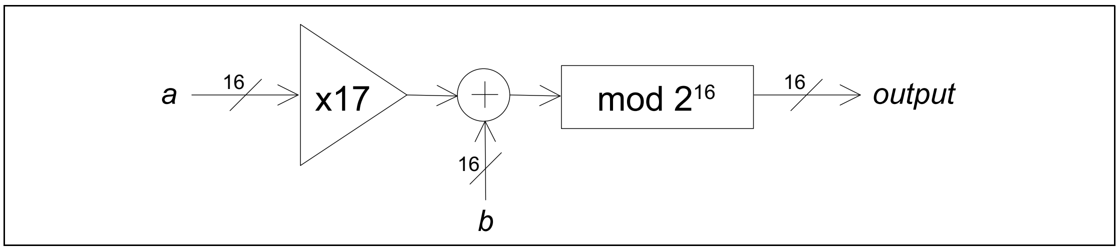


Figure 4.35: Multiply, Add, and Modulo block operation

The output of the MAM operation, given inputs *a* and *b*, is:

$$output = (17 \times a + b) \bmod 2^{16}$$

A *remapping table* is built that contains all the *used channels* in ascending order, indexed from zero.

4.5.8.3.3 Unmapped Event Channel Selection

The unmapped event channel selection process consists of two stages. First, the unsigned pseudo-random number *prn_e* is generated, after which the unmapped channel index *unmappedChannel* is derived from *prn_e*.

The first stage shall be as shown in Figure 4.36.

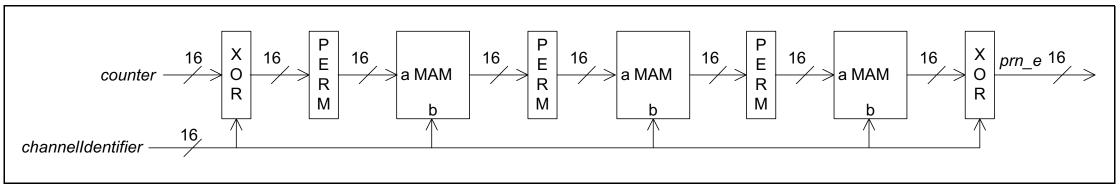


Figure 4.36: Event pseudo-random number generation

unmappedChannel is then calculated as *prn_e* modulo 37. A block diagram of the overall process is shown in Figure 4.37.

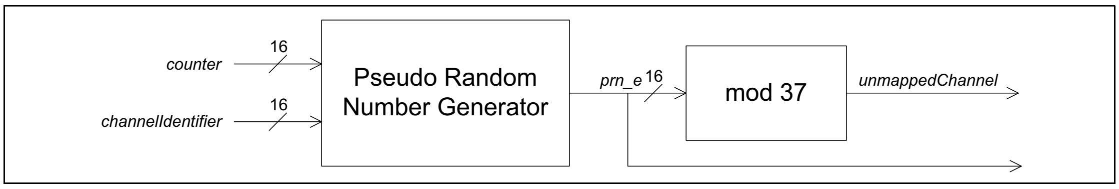


Figure 4.37: Unmapped channel selection process

4.5.8.3.4 Event Mapping to Used Channel Index

If *unmappedChannel* is the channel index of a *used channel* according to the channel map, it is used as the channel index for the event. If *unmappedChannel* is the index of an *unused channel* according to the channel map, then the channel index for the event is calculated from *prn_e* and *N* (the number of used channels) by first calculating the value *remappingIndex* as:

$$remappingindex = \left\lfloor \left(\frac{N * prn\_e}{2^{16}} \right) \right\rfloor$$

and then using *remappingIndex* as an index into the remapping table to obtain the channel index for the event.

The overall process is illustrated in Figure 4.38.

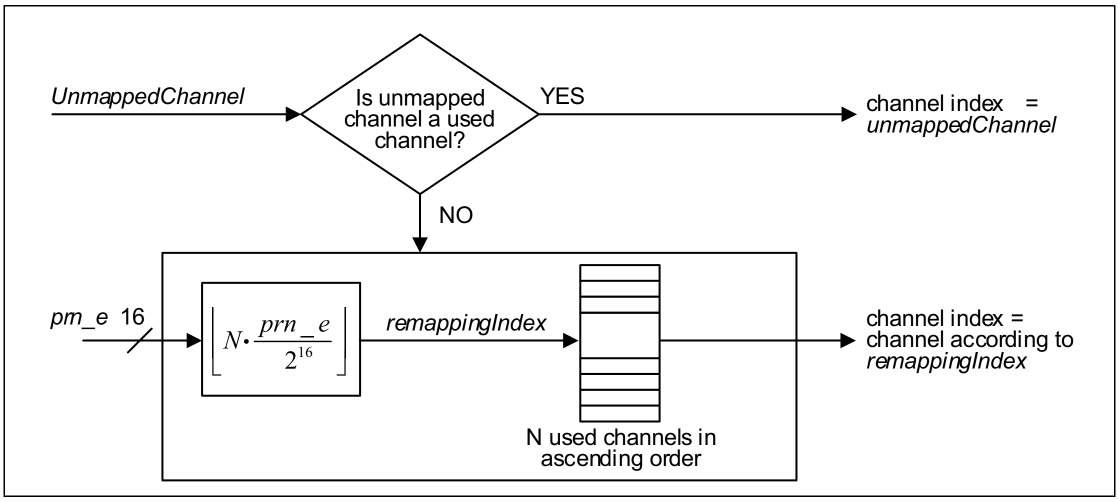

UnmappedChannel
Is unmapped channel a used channel?
YES
channel index = unmappedChannel
NO
prn_e 16
$\left\lfloor N \cdot \frac{prn\_e}{2^{16}} \right\rfloor$
remappingIndex
channel index = channel according to remappingIndex
N used channels in ascending order

Figure 4.38: Event mapping to used channel index process

4.5.9 Acknowledgment and Flow Control

The Link Layer acknowledgment 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. The *transmitSeqNum* parameter is used to identify packets sent by the Link Layer. The *nextExpectedSeqNum* parameter 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.

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

If the last Data Channel PDU was sent on the LE Coded PHY, the coding scheme (see Section 2.2.3) used when resending may be the same as or different from that used in the last Data Channel PDU. If the instant of a PHY

Update procedure (see Section 5.1.10) occurs while a Data Channel PDU is waiting to be resent, the new PHY shall be used when resending.

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 *nextExpectedSeqNum*. 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 4.39.

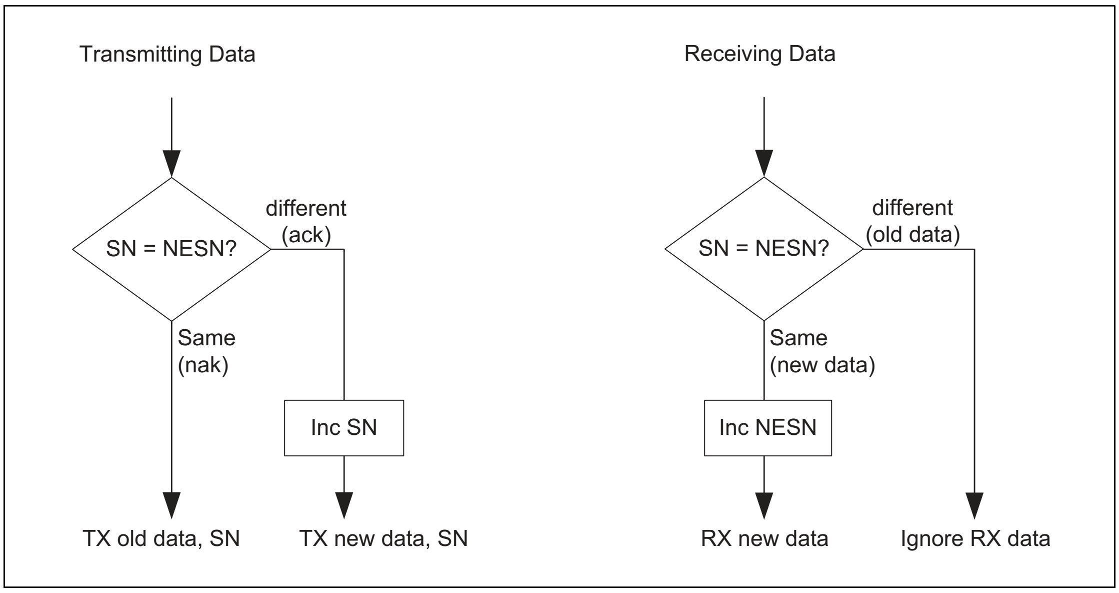


Figure 4.39: Transmit and Receive SN and NESN flow diagram

If a Data Channel PDU is received with an invalid CRC match, *nextExpectedSeqNum* 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 fail to 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.

4.5.10 Data PDU Length Management

The Controller shall maintain the following global parameters:

- *connInitialMaxTxOctets* - the value of *connMaxTxOctets* that the Controller will use for a new connection.
- *connInitialMaxTxTime* - a value that the Controller will use to determine the value of *connMaxTxTime* that it will use for a new connection.
- *connInitialMaxTxTimeUncoded* - the value of *connMaxTxTime* that the Controller will use for a new connection on an LE Uncoded PHY. The value of *connInitialMaxTxTimeUncoded* shall be the greater of 328 and the value of *connInitialMaxTxTime*.
- *connInitialMaxTxTimeCoded* - the value of *connMaxTxTime* that the Controller will use for a new connection on the LE Coded PHY. The value of *connInitialMaxTxTimeCoded* shall be the greater of 2704 and the value of *connInitialMaxTxTime*.
- *supportedMaxTxOctets* - the maximum value of *connMaxTxOctets* that the Controller supports.
- *supportedMaxTxTime* - the maximum value of *connMaxTxTime* that the Controller supports.
- *supportedMaxRxOctets* - the maximum value of *connMaxRxOctets* that the Controller supports.
- *supportedMaxRxTime* - the maximum value of *connMaxRxTime* that the Controller supports.

Note: 2704 μs is derived from the duration of a packet with a 27 octet payload when sent on the LE Coded PHY using S=8 coding.

The Controller shall maintain the following parameters for each connection:

- *connMaxTxOctets* - the maximum number of octets in the Payload field that the local device will send to the remote device.
- *connMaxRxOctets* - the maximum number of octets in the Payload field that the local device is able to receive from the remote device.
- *connRemoteMaxTxOctets* - the maximum number of octets in the Payload field that the remote device will send to the local device.
- *connRemoteMaxRxOctets* - the maximum number of octets in the Payload field that the remote device is able to receive from the local device.
- *connMaxTxTime* - the maximum number of microseconds that the local device will take to transmit a PDU to the remote device.
- *connMaxRxTime* - the maximum number of microseconds that the local device can take to receive a PDU from the remote device.
- *connRemoteMaxTxTime* - the maximum number of microseconds that the remote device will take to transmit a PDU to the local device.
- *connRemoteMaxRxTime* - the maximum number of microseconds that the remote device can take to receive a PDU from the local device.

The values of these parameters shall each be within the range given in Table 4.3:

LE Data Packet Length Extension feature supported	LE Coded PHY feature supported	Parameters with names ending in "Octets"		Parameters with names ending in "Time"	
		Minimum	Maximum	Minimum	Maximum
No	No	27	27	328	328
Yes	No	27	251	328	2120
No	Yes	27	27	328	2704
Yes	Yes	27	251	328	17040

Table 4.3: Valid ranges for data PDU length management parameters

The following values are derived from the parameters maintained by the Controller:

- *connEffectiveMaxTxOctets* - the lesser of *connMaxTxOctets* and *connRemoteMaxRxOctets*.
- *connEffectiveMaxRxOctets* - the lesser of *connMaxRxOctets* and *connRemoteMaxTxOctets*.
- *connEffectiveMaxTxTimeUncoded* - the lesser of *connMaxTxTime* and *connRemoteMaxRxTime*.
- *connIntervalPortionAvailable* - the current *connInterval* for the connection minus C, where:
  C = (2*T_IFS) + min (*connEffectiveMaxRxTime*, ((*connEffectiveMaxRxOctets* * 64) + 976))

Note: 976 µs is the duration of a packet with a zero octet payload when sent on the LE Coded PHY using S=8 coding.

- *connEffectiveMaxTxTimeAvailable* - the lesser of *connEffectiveMaxTxTimeUncoded* and *connIntervalPortionAvailable*.
- *connEffectiveMaxTxTimeCoded* - the greater of 2704 and *connEffectiveMaxTxTimeAvailable*.
- *connEffectiveMaxTxTime* - equal to *connEffectiveMaxTxTimeUncoded* while the connection is on an LE Uncoded PHY and equal to *connEffectiveMaxTxTimeCoded* while the connection is on the LE Coded PHY.
- *connEffectiveMaxRxTimeUncoded* - the lesser of *connMaxRxTime* and *connRemoteMaxTxTime*.
- *connEffectiveMaxRxTimeCoded* - the greater of 2704 and *connEffectiveMaxRxTimeUncoded*.
- *connEffectiveMaxRxTime* - equal to *connEffectiveMaxRxTimeUncoded* while the connection is on an LE Uncoded PHY and equal to *connEffectiveMaxRxTimeCoded* while the connection is on the LE Coded PHY.

Note: Corresponding octet and time parameters do not have to be mutually consistent. For example, it is permissible for a time parameter to be 2120 µs even though, on some PHYs, the maximum possible time is less.

The Controller shall not change the values of *supportedMaxTxOctets*, *supportedMaxTxTime*, *supportedMaxRxOctets*, and *supportedMaxRxTime*.

For a new connection:

- *connMaxTxOctets* shall be set to *connInitialMaxTxOctets* and *connMaxRxOctets* shall be chosen by the Controller. If either value is not 27 then the Controller should initiate the Data Length Update Procedure (Section 5.1.9) at the earliest practical opportunity.
- *connRemoteMaxTxOctets* and *connRemoteMaxRxOctets* shall be 27.

For a new connection on an LE Uncoded PHY:

- *connMaxTxTime* shall be set to *connInitialMaxTxTimeUncoded* and *connMaxRxTime* shall be chosen by the Controller. If either value is not 328, then the Controller should initiate the Data Length Update Procedure (Section 5.1.9) at the earliest practical opportunity.
- *connRemoteMaxTxTime* and *connRemoteMaxRxTime* shall be 328.

For a new connection on the LE Coded PHY:

- *connMaxTxTime* shall be set to *connInitialMaxTxTimeCoded* and *connMaxRxTime* shall be chosen by the Controller. If either value is not

2704, then the Controller should initiate the Data Length Update Procedure (Section 5.1.9) at the earliest practical opportunity.

- *connRemoteMaxTxTime* and *connRemoteMaxRxTime* shall be 2704.

The Controller may change the values of *connMaxTxOctets, connMaxRxOctets, connMaxTxTime,* and *connMaxRxTime* at any time after entering the Connection State. Whenever it does so, it shall communicate these values to the peer device using the Data Length Update Procedure. The values shall not exceed the values of *supportedMaxTxOctets*, *supportedMaxTxTime*, *supportedMaxRxOctets*, and *supportedMaxRxTime* respectively.

The Controller shall not transmit packets as part of a connection that have a maximum Payload Length greater than *connEffectiveMaxTxOctets* or that take more than *connEffectiveMaxTxTime* microseconds to transmit except during the period where the values of either *connEffectiveMaxTxOctets* or *connEffectiveMaxTxTime* are being modified. During that period, the Controller may still have Data Channel PDUs queued for transmission that conformed to the old parameters but violate the new ones. These PDUs remain valid; only PDUs queued after the Data Length Update Procedure is completed are required to conform to the changed parameters. However, a Controller should ensure that it has no Data Channel PDUs queued for transmission when it transmits an LL_LENGTH_REQ or LL_LENGTH_RSP PDU.

If the Controller decreases the value of *connMaxRxOctets* or *connMaxRxTime*, it shall not apply the new values until a Data Length Update Procedure (Section 5.1.9) that sends the new value has completed.

The Controller shall notify its Host if any of the parameters *connEffectiveMaxTxOctets*, *connEffectiveMaxRxOctets*, *connEffectiveMaxTxTime*, or *connEffectiveMaxRxTime* have changed.

4.6 FEATURE SUPPORT

The set of features supported by a Link Layer is represented by a bit mask called FeatureSet. The value of FeatureSet shall not change while the Controller has a connection to another device. A peer device may cache information about features that the device supports. The Link Layer may cache information about features that a peer supports during a connection.

Within FeatureSet, 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.

A Link Layer shall not use a procedure that is not supported by the peer’s Link Layer. A Link Layer shall not transmit a PDU listed in the following subsections unless it supports at least one of the features that requires support for that PDU.

The bit positions for each Link Layer Feature shall be as shown in Table 4.4. This table also shows if these bits are valid between Controllers. If a bit is shown as not valid, using 'N', then this bit shall be ignored upon receipt by the peer Controller.

Bit position	**Link Layer Feature**	**Valid from Controller to Controller**
0	LE Encryption	Y
1	Connection Parameters Request Procedure	Y
2	Extended Reject Indication	Y
3	Slave-initiated Features Exchange	Y
4	LE Ping	N
5	LE Data Packet Length Extension	Y
6	LL Privacy	N
7	Extended Scanner Filter Policies	N
8	LE 2M PHY	Y
9	Stable Modulation Index - Transmitter	Y
10	Stable Modulation Index - Receiver	Y
11	LE Coded PHY	Y
12	LE Extended Advertising	N
13	LE Periodic Advertising	N
14	Channel Selection Algorithm #2	Y
15	LE Power Class 1	Y
16	Minimum Number of Used Channels Procedure	
All other values	Reserved for Future Use	

Table 4.4: 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)
- LL_PAUSE_ENC_REQ (Section 2.4.2.11)
- LL_PAUSE_ENC_RSP (Section 2.4.2.12)
- Encryption Start Procedure (Section 5.1.3.1)
- Encryption Pause Procedure (Section 5.1.3.2)

4.6.2 Connection Parameters Request Procedure

A Controller that supports Connection Parameters Request Procedure shall support the following sections within this document:

- LL_REJECT_EXT_IND (Section 2.4.2.18)
- LL_CONNECTION_PARAM_REQ (Section 2.4.2.16)
- LL_CONNECTION_PARAM_RSP (Section 2.4.2.17)
- Connection Parameters Request Procedure (Section 5.1.7)

4.6.3 Extended Reject Indication

A Controller that supports Extended Reject Indication shall support the following sections within this document:

- LL_REJECT_EXT_IND (Section 2.4.2.18)

4.6.4 Slave-initiated Features Exchange

A Controller that supports Slave-initiated Features Exchange shall support the following sections within this document:

- LL_SLAVE_FEATURE_REQ (Section 2.4.2.15)
- LL_FEATURE_RSP (Section 2.4.2.10)

4.6.5 LE Ping

A Controller that supports LE Ping shall support the following sections of this document.

- LL_PING_REQ (Section 2.4.2.19)
- LL_PING_RSP (Section 2.4.2.20)
- LE Ping Procedure (Section 5.1.8)
- LE Authenticated Payload Timeout (Section 5.4)

4.6.6 LE Data Packet Length Extension

A Controller that supports LE Data Packet Length Extension shall support the following sections of this document.

- LL_LENGTH_REQ and LL_LENGTH_RSP (Section 2.4.2.21)
- Data Length Update Procedure (Section 5.1.9)

4.6.7 LL Privacy

A Controller that supports LL Privacy shall support the following sections of this document.

- LL Privacy (Section 6)

4.6.8 Extended Scanner Filter Policies

A Controller that supports Directed Advertising Report shall support the following sections of this document.

- Scanner Filter Policy (Section 4.3.3)

4.6.9 Multiple PHYs

A Controller that supports any PHY other than LE 1M PHY shall support the following sections within this document:

- Transmission and reception using the supported modulation schemes ([Vol 6] Part A, Section 1)
- LL_REJECT_EXT_IND Section 2.4.2.18)
- LL_PHY_REQ (Section 2.4.2.22)
- LL_PHY_RSP (Section 2.4.2.22)
- LL_PHY_UPDATE_IND (Section 2.4.2.23)
- PHY Update Procedure (Section 5.1.10)

4.6.9.1 Symmetric and Asymmetric Connections

A Controller shall support connections using the same PHY in each direction (“symmetric connections”) and may support connections using different PHYs in each direction (“asymmetric connections”).

If a Controller cannot support asymmetric connections then, in the PHY Update Procedure:

- Any LL_PHY_REQ or LL_PHY_RSP PDUs sent shall indicate that it wants a symmetric connection.
- Any LL_PHY_UPDATE_IND PDU sent shall not specify an asymmetric connection.

4.6.10 Stable Modulation Index - Transmitter

A Controller that supports Stable Modulation Index - Transmitter shall support the following section within this document:

- Stable Modulation Index ([Vol 6] Part A, Section 3.1.1)

4.6.11 Stable Modulation Index - Receiver

A Controller that supports Stable Modulation Index - Receiver shall support the following section within this document:

- Stable Modulation Index ([Vol 6] Part A, Section 4.7)

4.6.12 LE Extended Advertising

A Controller that supports LE Extended Advertising shall support reception of an Advertising Channel PDU payload of 255 octets and support the following sections of this document.

- ADV_EXT_IND (Section 2.3.1.5)
- AUX_ADV_IND (Section 2.3.1.6)
- AUX_CHAIN_IND (Section 2.3.1.8)
- AUX_SCAN_REQ (Section 2.3.2.1)
- AUX_SCAN_RSP (Section 2.3.2.3)
- AUX_CONNECT_REQ (Section 2.3.3.1)
- AUX_CONNECT_RSP (Section 2.3.3.2)
- Common Extended Advertising Payload Format (Section 2.3.4)
- Connectable Directed Event Type using ADV_EXT_IND (Section 4.4.2.4.4)
- Scannable Undirected Event Type using ADV_EXT_IND (Section 4.4.2.5.2)
- Connectable Undirected Event Type (Section 4.4.2.7)

- Scannable Directed Event Type (Section 4.4.2.8)
- Non-Connectable and Non-Scannable Directed Event Type (Section 4.4.2.9)
- Advertising Data Sets (Section 4.4.2.10)
- Using AdvDataInfo (ADI) (Section 4.4.2.11)
- Advertising Data Sets (Section 4.4.3.3)
- Connect Requests on the Secondary Advertising Channel (Section 4.4.4.2)

A Controller that supports connections shall also support the Channel Selection Algorithm #2 feature.

4.6.13 LE Periodic Advertising

A Controller that supports LE Periodic Advertising shall support the LE Extended Advertising feature, Channel Selection Algorithm #2 feature, and the following sections of this document.

- AUX_SYNC_IND (Section 2.3.1.7)
- Periodic Advertising (Section 4.4.2.12 and Section 4.4.3.4)

4.6.14 Channel Selection Algorithm #2

A Controller that supports Channel Selection Algorithm #2 shall support the following sections within this document:

- ChSel bit set to 1 (Sections 2.3, 2.3.1.1, 2.3.1.2, and 2.3.3.1)
- Channel Selection Algorithm #2 (Section 4.5.8.3).

4.6.15 Minimum Number of Used Channels Procedure

A Controller that supports the Minimum Number of Used Channels Procedure shall support the following sections of this document:

- LL_MIN_USED_CHANNELS_IND (Section 2.4.2.24)
- Minimum Number Of Used Channels Procedure (Section 5.1.11)

4.7 RESOLVING LIST

All Link Layers supporting Link Layer Privacy (see Section 6) shall contain a set of records for local and peer IRK value pairs. These values are known as the Local IRK and the Peer IRK. The Resolving List IRK pairs shall be associated with a public or static device address known as the Identity Address. The Identity Address may be in the White List. All Link Layers supporting Link Layer Privacy shall support a Resolving List capable of storing at least one Resolving List Record.

On reset, the Resolving List shall be empty.

The Resolving List is configured by the Host and is used by the Link Layer to resolve Resolvable Private Addresses used by advertisers, scanners or initiators. This allows the Host to configure the Link Layer to act on a request without awakening the Host.

The White List and filter policies set by the Host are applied to the associated Identity Address once the Resolvable Private Address has been resolved.

If the Host, when populating the resolving list, sets a peer IRK to all zeros, then the peer address used within an advertising channel PDU shall use the peer's Identity Address, which is provided by the Host.

The Host specifies the privacy mode to be used with each peer identity on the resolving list. If it specifies that device privacy mode is to be used, then the Controller shall accept both the peer's device identity address and a resolvable private address generated by the peer device using its distributed IRK. Otherwise, network privacy mode is used: the Controller shall only accept resolvable private addresses generated by the peer device using its distributed IRK. If the Host has added the peer device to the resolving list with an all-zero peer IRK, the Controller shall only accept the peer's identity address, as defined in Section 6.5.

If the Host, when populating the resolving list, sets a local IRK to all zeros, then any local address used within an advertising channel PDU shall use the local Identity Address, which is provided by the Host.

If the Link Layer is using the Resolving List and the peer device has been resolved, the Address returned to the Host is the peer device's Identity Address.

If the Link Layer is using the Resolving List and the peer device has been resolved but the encryption fails then the current Resolvable Private Address(es) shall be immediately discarded and new Resolvable Private Address(es) shall be generated.

Note: Encryption may fail when the address was resolved successfully using an incorrect IRK and, therefore, encryption keys on both sides did not match.

When the Controller address resolution is enabled, both peer and local RPAs received by the Link Layer shall be resolved using the Resolving List.

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. However, except where prohibited elsewhere in this section, a Link Layer may initiate an LL Control Procedure while responding to a procedure initiated by its peer device.

There are no restrictions on the order that Link Layer Control Procedures are carried out except that no procedure can be started until after entering the Connection State; i.e., no procedure requires a different procedure to be carried out previously.

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 may initiate the update of the connection parameters by sending an LL_CONNECTION_UPDATE_IND PDU if either the master or slave or both do not support the Connection Parameters Request procedure (Section 5.1.7). The slave shall not send this PDU. The slave may request a change to the connection parameters using the L2CAP LE signaling channel if either the master or the slave or both do not support the Connection Parameters Request procedure (Section 5.1.7). The slave may request a change to the connection parameters using the Connection Parameters Request Procedure if both the master and slave support the Connection Parameters Request procedure.

In order to request a change to the connection parameters, the master shall use the Connection Parameters Request Procedure if both the master and slave support that procedure. If the slave rejects the Connection Parameters

Request Procedure, then the master may update the connection parameters using the Connection Update Procedure.

The Link Layer of the master shall determine the *connInterval* from the interval range given by the Host ($\mathit{connInterval}_{min}$ and $\mathit{connInterval}_{max}$). However, if the current PHY is the LE Coded PHY and the Controller supports the LE Data Packet Length Extension feature, then the new connection interval shall be at least C µs, where:

C = (2*T_IFS) + min (*connEffectiveMaxRxTime*, ((*connEffectiveMaxRxOctets* * 64) + 976)) + 2704

Note: 976 µs and 2704 µs are derived from the durations of packets with a zero octet payload and with a 27 octet payload when sent on the LE Coded PHY using S=8 coding.

The Link Layer shall indicate to the Host the selected interval value.

Section 5.5 shall apply to the LL_CONNECTION_UPDATE_IND PDU. When the slave receives such a PDU with the instant in the future, it shall listen to the connection event where connEventCount equals Instant and the connection event before it.

The connection interval used before the instant is known as $\mathit{connInterval}_{OLD}$. The connection interval contained in the LL_CONNECTION_UPDATE_IND PDU and used at the instant and after, is known as $\mathit{connInterval}_{NEW}$.

The connection slave latency used before the instant is known as $\mathit{connSlaveLatency}_{OLD}$. The connection slave latency contained in the LL_CONNECTION_UPDATE_IND PDU and used at the instant and after, is known as $\mathit{connSlaveLatency}_{NEW}$.

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

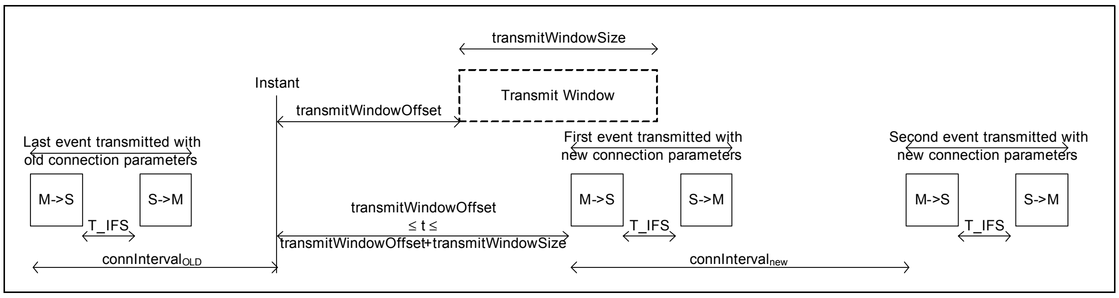

transmitWindowSize
Instant
Transmit Window
transmitWindowOffset
Last event transmitted with old connection parameters
First event transmitted with new connection parameters
Second event transmitted with new connection parameters
M->S
S->M
T_IFS
transmitWindowOffset ≤ t ≤ transmitWindowOffset+transmitWindowSize
connInterval$_{OLD}$
connInterval$_{new}$

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

For example, the interval between the preceding connection event before the instant and the instant will be $connInterval_{OLD}$. The interval between the connection event after the instant and the following connection event will be $connInterval_{NEW}$.

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 after 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 instant occurs after $connInterval_{OLD}$ and before *transmitWindowOffset*. All the normal connection event transmission rules specified in Section 4.5.1, shall apply.

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_IND 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_IND or CONNECT_IND 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_IND PDU. The slave shall not send this PDU. The master Controller can update the channel map without being requested to by the Host.

Section 5.5 shall apply to the LL_CHANNEL_MAP_IND PDU.

The channel map used before the instant is known as $channelMap_{OLD}$. The channel map contained in the LL_CHANNEL_MAP_IND PDU and used at the instant and after, is known as $channelMap_{NEW}$.

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_IND 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). IVm shall be a 32 bit random number generated by the Link Layer of the master. SKDm shall be a 64 bit random number generated by the Link Layer of the master. Both IVm and SKDm shall be generated using the requirements for random number generation defined in [Vol 2] Part H, Section 2.

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_RSP, LL_TERMINATE_IND, LL_REJECT_IND or LL_REJECT_EXT_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 or LL_REJECT_EXT_IND PDU with the ErrorCode set to *Unsupported Remote Feature / Unsupported LMP Feature* (0x1A). The Link Layer of the master receiving the LL_REJECT_IND or LL_REJECT_EXT_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 or LL_REJECT_EXT_IND PDU from the slave. The procedure is complete in the slave when the acknowledgment for the LL_REJECT_IND or LL_REJECT_EXT_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). IVs shall be a 32 bit random number generated by the Link Layer of the slave. SKDs shall be a 64 bit random number generated by the Link Layer of the slave. Both IVs and SKDs shall be generated using the requirements for random number generation defined in [Vol 2] Part H, Section 2.

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, LL_REJECT_IND or LL_REJECT_EXT_IND PDUs.

The Link Layer of the slave shall then send an LL_ENC_RSP PDU. The Link Layer of the slave shall then notify the Host with the Rand and EDIV fields. After having sent the LL_ENC_RSP PDU, the Link Layer of the slave can receive an LL_UNKNOWN_RSP PDU corresponding to a LL Control PDU sent by the slave. The slave should not disconnect the link in this case.

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* (0x06).
- 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 either send an LL_REJECT_IND with the ErrorCode set to *PIN or key Missing* (0x06) or an LL_REJECT_EXT_IND PDU with the RejectOpcode set to "LL_ENC_REQ" and the ErrorCode set to *PIN or key Missing* (0x06). Upon receiving an LL_REJECT_IND or LL_REJECT_EXT_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 or LL_REJECT_EXT_IND PDU from the slave. The procedure is completed in the slave when the acknowledgment has been received for the LL_REJECT_IND or LL_REJECT_EXT_IND PDU from the master.
- If the Host does provide a Long Term Key, the Link Layer of the slave shall calculate *sessionKey* using the encryption engine with LTK as the key, and SKD as the plain text input. The *sessionKey* parameter 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.

If at any time during the encryption start procedure, the Link Layer of the master or the slave receives an unexpected Data Channel PDU from the peer Link Layer, it shall immediately exit the Connection State, and shall transition to the Standby State. The Host shall be notified that the link has been disconnected with the error code *Connection Terminated Due to MIC Failure* (0x3D).

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.

If at any time during the encryption pause procedure, the Link Layer of the master or the slave receives an unexpected Data Channel PDU from the peer Link layer, it shall immediately exit the Connection State, and shall transition to the Standby State. The Host shall be notified that the link has been disconnected with the error code *Connection Terminated Due to MIC Failure* (0x3D).

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. Both the master and slave can initiate this procedure.

The FeatureSet information may be cached either during a connection or between connections. 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 from a previous connection is not 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.

The FeatureSet_M parameter is the feature capabilities of the Link Layer of the master.

The FeatureSet_S parameter is the feature capabilities of the Link Layer of the Slave.

The FeatureSet_{USED} parameter is one octet long and is the logical AND of $\text{FeatureSet}_M[0]$ and $\text{FeatureSet}_S[0]$.

5.1.4.1 Master-initiated Feature Exchange Procedure

The master initiates this procedure with an LL_FEATURE_REQ PDU, and the slave responds with an LL_FEATURE_RSP PDU.

When the Link Layer of the master sends an LL_FEATURE_REQ PDU, the FeatureSet field shall be set to FeatureSet_{M}.

When the Link Layer of the slave sends an LL_ FEATURE_RSP PDU, octet 0 of the FeatureSet field shall be set to FeatureSet_{USED} and the remaining octets shall be set to the corresponding octets of FeatureSet_{S}.

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

5.1.4.2 Slave-initiated Feature Exchange Procedure

The slave initiates this procedure with an LL_SLAVE_FEATURE_REQ PDU, and the master responds with an LL_FEATURE_RSP PDU.

When the Link Layer of the slave sends an LL_SLAVE_FEATURE_REQ PDU, the FeatureSet field shall be set to FeatureSet_{S}.

When the Link Layer of the master sends an LL_FEATURE_RSP PDU, octet 0 of the FeatureSet field shall be set to FeatureSet_{USED} and the remaining octets shall be set to the corresponding octets of FeatureSet_{M}.

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

When the Link Layer of the master receives an LL_SLAVE_FEATURE_REQ PDU, it shall send an LL_FEATURE_RSP PDU.

If the Link Layer of the slave sends the LL_SLAVE_FEATURE_REQ PDU to a master that does not understand that PDU, then the slave should expect an LL_UNKNOWN_RSP PDU in response. If the LL_SLAVE_FEATURE_REQ PDU was issued as a result of a Host-initiated read remote features procedure (see [Vol 2] Part E, Section 7.8.21), then the Host shall be notified that the read remote features procedure has completed with the ErrorCode set to *Unsupported Remote Feature / Unsupported LMP Feature* (0x1A).

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

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 acknowledgment is received or until the timer, $T_{terminate}$, expires, after which it shall exit the Connection State and transition to the Standby State. 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 acknowledgment, exit the Connection State and shall transition to the Standby State.

The procedure has completed when the acknowledgment has been received or the timer, $T_{terminate}$, expires.

5.1.7 Connection Parameters Request Procedure

The master or slave may initiate a Connection Parameters Request procedure to request the remote device to have the Link Layer parameters for the connection (*connInterval*, *connSlaveLatency* and *connSupervisionTimeout*) updated any time after entering the Connection State.

5.1.7.1 Issuing an LL_CONNECTION_PARAM_REQ PDU

The Connection Parameters Request procedure is initiated by issuing an LL_CONNECTION_PARAM_REQ PDU. The procedure can be initiated as a result of a Host initiated connection update procedure (see [Vol 2] Part E, Section 7.8.18) or autonomously by the Link Layer (that is, without being requested by the Host).

If the Link Layer of the master or slave sends the LL_CONNECTION_PARAM_REQ PDU to a device that does not understand that PDU, then the device should expect an LL_UNKNOWN_RSP PDU in response. If the LL_CONNECTION_PARAM_REQ PDU was issued by the Link Layer of the slave as a result of a Host initiated connection update procedure, then the Host shall be notified that the connection update procedure has completed with the ErrorCode set to *Unsupported Remote Feature / Unsupported LMP Feature* (0x1A).

If the Link Layer initiates this procedure as a result of a Host initiated connection update procedure, then the Link Layer:

- Should set the Interval_Min, Interval_Max, Timeout, and Latency fields to the values received from the Host. Note: The Link Layer may modify the values of these fields, for example, because the values received from the Host would prevent the Link Layer from meeting commitments in another piconet.
- May indicate the preferred periodicity by setting the PreferredPeriodicity field to a value other than zero, as described in Section 2.4.2.16.
- May set the Offset0 to Offset5 fields to a value other than 0xFFFF as described in Section 2.4.2.16. If one or more of the Offset0 to Offset5 fields have been set, then:
  - The ReferenceConnEventCount field shall be set to indicate that at least one of the Offset0 to Offset5 fields is valid. If the ReferenceConnEvent-Count field is set, then it shall always be set to the connEventCount of a connection event that is less than 32767 connection events in the future from the first transmission of the PDU. Note: Retransmissions of the PDU can result in the ReferenceConnEventCount to be up to 32767 events in the past when the PDU is successfully received by the remote device. See Section 5.1.7.3.2 for examples on how to set the ReferenceConnE-ventCount field.
  - If Interval_Min is not equal to Interval_Max then the PerferredPeriodicity field shall be set to a value other than zero. If Interval_Min is equal to

Interval_Max then the PreferredPeriodicity field may be set to any value and shall be ignored by the recipient.

If the Link Layer initiates this procedure autonomously, then the Latency field shall be set to the current value of *connSlaveLatency* and the Timeout field (in milliseconds) shall be set to the current value of *connSupervisionTimeout*. Any of the other fields (Interval_Min, Interval_Max, PreferredPeriodicity, ReferenceConnEventCount and Offset0 to Offset5) may be changed within the restrictions given above.

The Link Layer shall ensure that the parameters in the LL_CONNECTION_PARAM_REQ shall not cause supervision timeout. That is, the Link Layer shall ensure that the Timeout (in milliseconds) is greater than 2* Interval_Max * (Latency + 1).

5.1.7.2 Responding to LL_CONNECTION_PARAM_REQ and LL_CONNECTION_PARAM_RSP PDUs

Upon receiving an LL_CONNECTION_PARAM_REQ PDU:

- The slave shall respond with either an LL_CONNECTION_PARAM_RSP PDU or an LL_REJECT_EXT_IND PDU.
- The master shall respond with either an LL_CONNECTION_UPDATE_IND PDU or an LL_REJECT_EXT_IND PDU.

Upon receiving an LL_CONNECTION_PARAM_RSP PDU, the master shall respond with either an LL_CONNECTION_UPDATE_IND PDU or an LL_REJECT_EXT_IND PDU.

The master shall not send the LL_CONNECTION_PARAM_RSP PDU. The slave shall send an LL_CONNECTION_PARAM_RSP PDU only in response to an LL_CONNECTION_PARAM_REQ PDU.

If the received LL_CONNECTION_PARAM_REQ PDU contains parameters that are not acceptable to the Link Layer, then the Link Layer of the device shall respond to the LL_CONNECTION_PARAM_REQ PDU with one of the following:

- An LL_CONNECTION_PARAM_RSP PDU (if the Link Layer is the slave of the connection) or an LL_CONNECTION_UPDATE_IND PDU (if the Link Layer is the master of the connection) containing alternative parameters.
- An LL_REJECT_EXT_IND PDU with the ErrorCode set to *Unsupported LL Parameter Value* (0x20).

If the received LL_CONNECTION_PARAM_REQ PDU contains any fields that are out of valid range, then the Link Layer shall reject the LL_CONNECTION_PARAM_REQ PDU by issuing an LL_REJECT_EXT_IND PDU with the ErrorCode set to *Invalid LL Parameters* (0x1E).

If an LL_REJECT_EXT_IND PDU is sent during the Connection Parameters Request procedure, then the procedure is complete on a device when it receives the LL_REJECT_EXT_IND PDU, and is complete on the device that issued the LL_REJECT_EXT_IND PDU when it receives the acknowledgment for the LL_REJECT_EXT_IND PDU.

If the received LL_CONNECTION_PARAM_REQ PDU requests only a change in the anchor points of the LE connection, then the Link Layer shall not indicate this request to its Host.

If the received LL_CONNECTION_PARAM_REQ PDU requests a change to one or more of *connInterval*, *connSlaveLatency*, and *connSupervisionTimeout* and if the values selected by the Link Layer are, respectively, within the range of the *connInterval*, the value of *connSlaveLatency* and the value of *connSupervisionTimeout* provided by the local Host, then the Link Layer may choose to not indicate this request to its Host and proceed as if the Host has accepted the remote device's request. Otherwise, if the event to the Host is not masked, then the Link Layer shall first indicate this request to its Host.

If the local Host has not provided the range of *connInterval*, the value of *connSlaveLatency* and the value of *connSupervisionTimeout* to the Link Layer of the slave, then the Link Layer of the slave may indicate the received request to its Host if the event to the Host is not masked.

If the request is being indicated to the Host and the event to the Host is masked, then the Link Layer shall issue an LL_REJECT_EXT_IND PDU with the ErrorCode set to *Unsupported Remote Feature / Unsupported LMP Feature* (0x1A). Note: The device could have issued the LL_REJECT_EXT_IND PDU temporarily. The initiating device may retry. Note: If the request is not being indicated to the Host, then the event mask is ignored.

If the Host is indicated of the request, it shall either accept or reject this request. If the Host rejects this request, then the device shall issue an LL_REJECT_EXT_IND PDU with the ErrorCode set to a value provided by the Host. The Host shall only use the error code *Unacceptable Connection Parameters* (0x3B) in order to reject the request.

If the Host accepts this request or if the request was not indicated to the Host, then:

- The slave shall respond to an LL_CONNECTION_PARAM_REQ PDU with an LL_CONNECTION_PARAM_RSP PDU. The rules for filling in various fields of the LL_CONNECTION_PARAM_RSP PDU are the same as those for filling in various fields of the LL_CONNECTION_PARAM_REQ PDU, as described in Section 5.1.7.1. The rules for handling a received LL_CONNECTION_PARAM_RSP PDU on the Link Layer of the master are identical to the rules for handling a received LL_CONNECTION_PARAM_REQ PDU that are described earlier in this section.

- The master shall respond to an LL_CONNECTION_PARAM_REQ PDU or an LL_CONNECTION_PARAM_RSP PDU with an LL_CONNECTION_UPDATE_IND PDU. The master should try to choose a value of Interval that is a multiple of PreferredPeriodicity if the slave has set the PreferredPeriodicity field of the LL_CONNECTION_PARAM_REQ or LL_CONNECTION_PARAM_RSP PDU. The master should try to pick the values of WinOffset and WinSize such that the timing of the new connection events matches one of the Offset0 to Offset5 fields of the LL_CONNECTION_PARAM_REQ PDU or the LL_CONNECTION_PARAM_RSP PDU sent by the slave. The Instant field of the LL_CONNECTION_UPDATE_IND PDU is set as described in Section 5.1.1.

Once the master issues the LL_CONNECTION_UPDATE_IND PDU, the connection parameters get updated as described in Section 5.1.1.

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

5.1.7.3 Examples

5.1.7.3.1 Slave initiated anchor point move

The following example shows the Link Layer of the slave requesting a change in the anchor points of the LE connection by 3.75ms.

The Link Layer of the slave issues an LL_CONNECTION_PARAM_REQ PDU with the following parameters:

- Interval_Min: *connInterval*
- Interval_Max: *connInterval*
- Latency: *connSlaveLatency*
- Timeout: *connSupervisionTimeout*
- PreferredPeriodicity: 0
- ReferenceConnEventCount: <any value that is less than 32767 connection events in the future>
- Offset0: 0x0003
- Offset1: 0xFFFF
- Offset2: 0xFFFF
- Offset3: 0xFFFF
- Offset4: 0xFFFF
- Offset5: 0xFFFF

If the Link Layer of the master accepts the slave's request, then it could respond with an LL_CONNECTION_UPDATE_IND PDU that contains any one of the following set of parameters. In all the sets, Interval is set to *connInterval*, Latency is set to *connSlaveLatency*, Timeout is set to *connSupervisionTimeout* and Instant is set to any value that is less than 32767 connection events in the future.

- Option 1: the first packet sent after the instant by the master is inside the Transmit Window and 3.75ms from the beginning of the Transmit Window.
  - 3 ≤ WinSize ≤ 8
  - WinOffset: 0
- Option 2: the first packet sent after the instant by the master is inside the Transmit Window and 2.5ms from the beginning of the Transmit Window.
  - 2 ≤ WinSize ≤ 8
  - WinOffset: 1
- Option 3: the first packet sent after the instant by the master is inside the Transmit Window and1.25ms from the beginning of the Transmit Window.
  - 1 ≤ WinSize ≤ 8
  - WinOffset: 2
- Option 4: the first packet sent after the instant by the master is inside the Transmit Window and 0ms from the beginning of the Transmit Window.
  - 1 ≤ WinSize ≤ 8
  - WinOffset: 3

5.1.7.3.2 ReferenceConnEventCount

Figure 5.2 and Figure 5.3 show examples of how the ReferenceConnEventCount and the Offset0 to Offset5 fields of the LL_CONNECTION_PARAM_REQ and the LL_CONNECTION_PARAM_RSP PDU can be utilized to indicate the possible position of the anchor points of the connection with the new connection parameters relative to the anchor points of the connection with the old connection parameters. This figure only shows Offset0 (and not Offset1 to Offset5) for simplicity. The figure also shows the Instant where the updated connection parameters are applied. Note that the actual Instant occurs $connInterval_{OLD}$ after the last connection event transmitted with the old connection parameters whereas the Instant field in the LL_CONNECTION_UPDATE_IND PDU is set to the *connEventCount* of the connection event transmitted with the old connection parameters.

The ReferenceConnEventCount is set to the *connEventCount* of the connection event on the old connection parameters such that the start of the very next connection event on the new connection parameters is Offset0 (in milliseconds) away from the start of the ReferenceConnEventCount connection event.

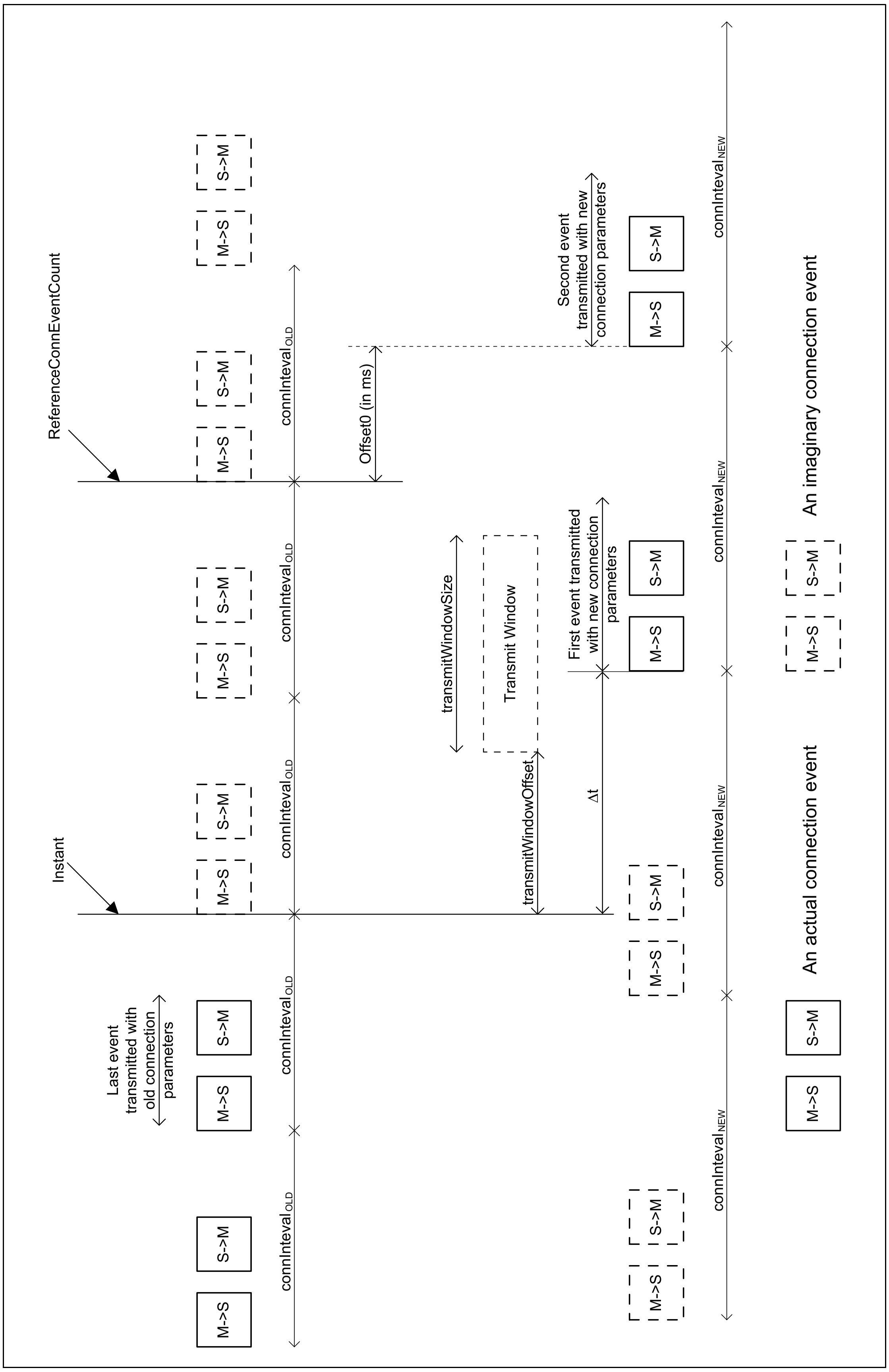
Figure 5.2 shows the case where the Instant is before the ReferenceConnEventCount. Figure 5.3 shows the case where the Instant is after the ReferenceConnEventCount. Imaginary connection events transmitted with the old connection parameters have been shown beyond the Instant and imaginary connection events transmitted with the new connection parameters have been shown before the Instant.

In Figure 5.2 and Figure 5.3, the time interval, Δt, between the Instant and the start of the first connection event transmitted with the new connection parameters can be calculated using the following equation:

$\Delta t = (connInterval_{NEW} - ((Instant - ReferenceConnEventCount) * connInterval_{OLD}) \% \; connInterval_{NEW} + offset0) \% \; connInterval_{NEW}$

Note: The case where the ReferenceConnEventCount and Instant are on different sides of the eventCount wraparound point is not shown in the equations above.

Based on the calculated Δt, the WinOffset and WinSize fields in the LL_CONNECTION_UPDATE_IND PDU could be set accordingly. See Section 5.1.7.3.3 for an example.



Instant
ReferenceConnEventCount
Last event transmitted with old connection parameters
M->S
S->M
$\text{connInteval}_{\text{OLD}}$
Offset0 (in ms)
transmitWindowSize
Transmit Window
transmitWindowOffset
Δt
First event transmitted with new connection parameters
Second event transmitted with new connection parameters
$\text{connInteval}_{\text{NEW}}$
An actual connection event
An imaginary connection event

Figure 5.2: Utilizing the ReferenceConnEventCount and Offset0 fields to indicate position of the new anchor points – Instant is before the ReferenceConnEventCount

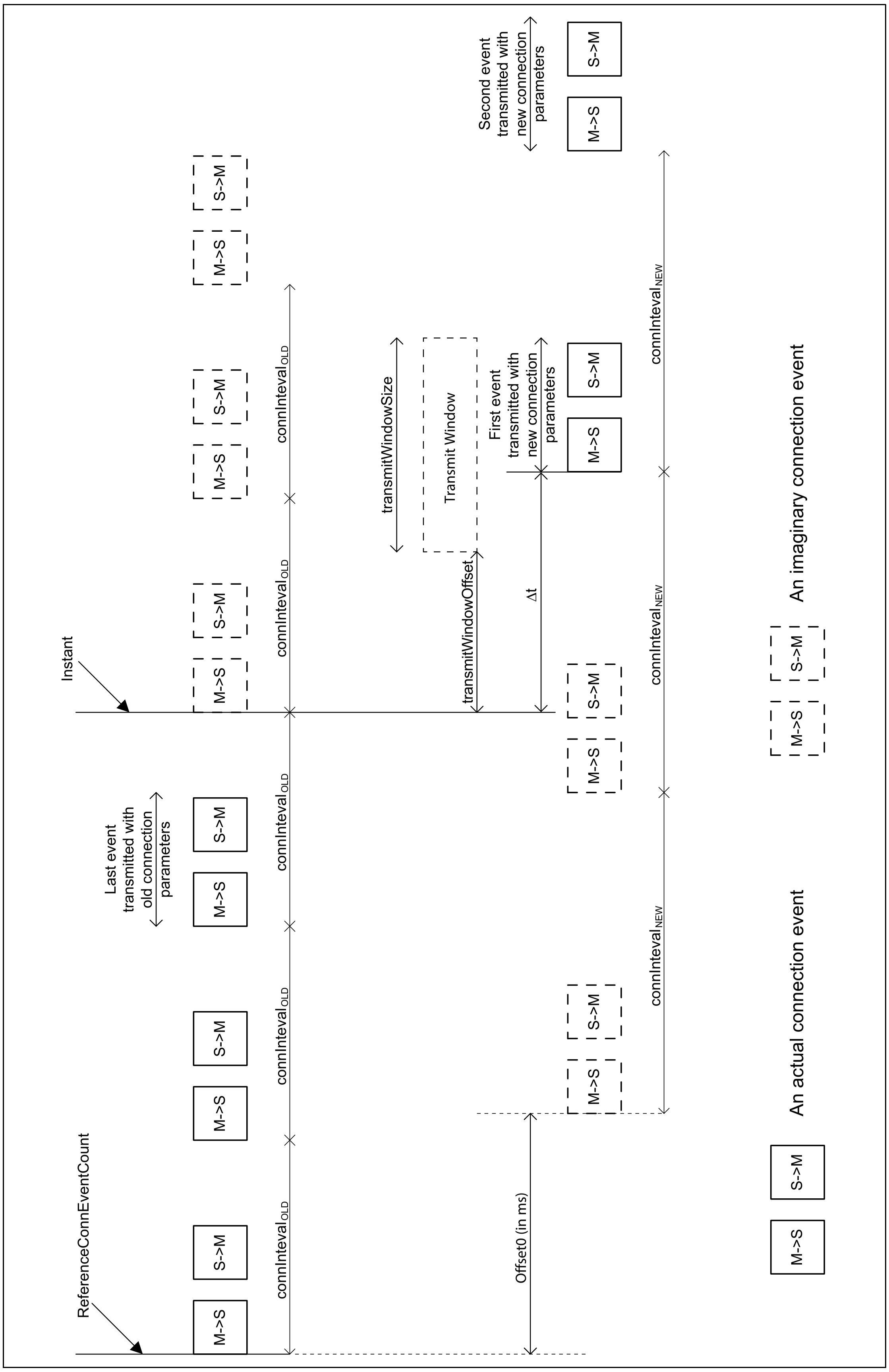

ReferenceConnEventCount
Instant
Last event transmitted with old connection parameters
M->S
S->M
$connInteval_{OLD}$
transmitWindowSize
Transmit Window
transmitWindowOffset
First event transmitted with new connection parameters
Second event transmitted with new connection parameters
Offset0 (in ms)
Δt
$connInteval_{NEW}$
An actual connection event
An imaginary connection event

Figure 5.3: Utilizing the ReferenceConnEventCount and Offset0 fields to indicate position of the new anchor points – Instant is after the ReferenceConnEventCount

5.1.7.3.3 Slave initiated interval and anchor point move

The following example shows the Link Layer of the slave requesting a change in both the connection interval (by indicating a PreferredPeriodicity such that PreferredPeriodicity and $connInterval_{OLD}$ are not integral multiples of one another) and a change in anchor points of the LE connection by 3.75ms with respect to the ReferenceConnEventCount.

In this example, $connInterval_{OLD}$ is 0x0C (15 ms). The Link Layer of the slave issues an LL_CONNECTION_PARAM_REQ PDU with the following parameters:

- Interval_Min: 0x16
- Interval_Max: 0x20
- Latency: *connSlaveLatency*
- Timeout: *connSupervisionTimeout*
- PreferredPeriodicity: 0x0A
- ReferenceConnEventCount: 0x1F00
- Offset0: 0x0003
- Offset1: 0xFFFF
- Offset2: 0xFFFF
- Offset3: 0xFFFF
- Offset4: 0xFFFF
- Offset5: 0xFFFF

If the Link Layer of the master accepts the slave's request, then it could respond with an LL_CONNECTION_UPDATE_IND PDU that contains any one of the following set of parameters. In all the sets, the new connection interval $connInterval_{NEW}$ is set to 0x1E (37.5 ms), Latency is set to *connSlaveLatency*, Timeout is set to *connSupervisionTimeout* and Instant is set to 0x1F06.

Δt, as described in Section 5.1.7.3.2 is calculated as 21 (26.25 ms).

The WinSize and WinOffset fields in the LL_CONNECTION_UPDATE_IND PDU can contain any of the following example set of parameters:

- Option 1: the first packet sent after the instant by the master is inside the Transmit Window and 3.75 ms from the beginning of the Transmit Window.
  - $3 \le$ WinSize ≤ 8
  - WinOffset: 18
- Option 2: the first packet sent after the instant by the master is inside the Transmit Window and 2.5 ms from the beginning of the Transmit Window.
  - $2 \le$ WinSize ≤ 8
  - WinOffset: 19

- Option 3: the first packet sent after the instant by the master is inside the Transmit Window and 1.25ms from the beginning of the Transmit Window.
  - $1 \leq$ WinSize ≤ 8
  - WinOffset: 20
- Option 4: the first packet sent after the instant by the master is inside the Transmit Window and 0ms from the beginning of the Transmit Window.
  - $1 \leq$ WinSize ≤ 8
  - WinOffset: 21

5.1.7.4 Packet Transmit Time Restrictions

This section only applies if the current PHY is the LE Coded PHY and the Controller supports the LE Data Packet Length Extension feature.

After having sent or received an LL_CONNECTION_UPDATE_IND PDU that decreases the connection interval, and until the instant has been reached, the Link Layer shall not transmit a packet that would take longer than *connEffectiveMaxTxTime* microseconds (see Section 4.5.10) to transmit, calculated using the connection interval that will apply after the instant.

After a slave sends an LL_CONNECTION_PARAM_REQ or LL_CONNECTION_PARAM_RSP PDU where *Interval_Min* indicates an interval less than the current connection interval, and until it receives an LL_CONNECTION_UPDATE_IND, LL_UNKNOWN_RSP, or LL_REJECT_EXT_IND PDU in response, its link layer shall not transmit a packet that would take longer than *connEffectiveMaxTxTime* microseconds to transmit, calculated using a connection interval corresponding to the *Interval_Min* value in the transmitted PDU.

If the value of *connEffectiveMaxTxTime* changes during the procedure, the above requirements apply to the value at the moment the Data Channel PDU is queued for transmission.

Note: The requirements of this section are in addition to, and do not override, those in Section 4.5.10.

Note: If a Link Layer has any Data Channel PDUs queued for transmission at the start of the procedure or queues any during the procedure, it may need to re-fragment those PDUs in order to meet these requirements.

5.1.8 LE Ping Procedure

The LE Ping procedure, when supported, can be used at the Link Layer to verify presence of the remote Link Layer. The procedure can also be used to verify message integrity on the LE ACL logical transport by forcing the remote device to send an LE ACL packet that contains a valid MIC.

Either the master or the slave Link Layer may initiate this procedure at any time after entering the Connection State by sending an LL_PING_REQ PDU. The responding Link Layer responds with the LL_PING_RSP PDU.

The Link Layer supporting this feature shall send an LL_PING_REQ PDU when the remote device has not sent a packet containing a payload protected by a MIC within the authenticated payload timeout set by the Host ([Vol 2] Part E, Section 7.3.94).

The procedure is completed when either an LL_PING_RSP is received or, in the case the remote device does not support the LE Ping feature, an LL_UNKNOWN_RSP is received with the Unknown type set to LL_PING_REQ.

5.1.9 Data Length Update Procedure

A Controller uses the Data Length Update Procedure to transmit the latest values of the current maximum Receive Data Channel PDU Payload Length and PDU Time (*connMaxRxOctets* and *connMaxRxTime*) and the current maximum Transmit Data Channel PDU Payload Length and PDU Time (*connMaxTxOctets* and *connMaxTxTime*) to the peer device.

Both the master and slave can initiate this procedure at any time after entering the Connection State by sending an LL_LENGTH_REQ PDU. This procedure shall be initiated by the Link Layer whenever any of these parameters change, whether requested by the Host or autonomously by the Link Layer. However, if this procedure has already been initiated by the remote Controller and the local Controller has not yet responded, it shall use the response to communicate the changes instead of initiating a new procedure.

If the Link Layer receives an LL_LENGTH_REQ, or an LL_LENGTH_RSP PDU that was a response to an LL_LENGTH_REQ PDU, then it shall update its *connRemoteMaxTxOctets*, *connRemoteMaxRxOctets*, *connRemoteMaxTxTime*, and *connRemoteMaxRxTime* parameters for the connection with the values in the PDU. It shall immediately start using the updated values for all new Data Channel PDUs queued for transmission (including any response as specified in the next paragraph). The length of any Data Channel PDUs that have already been queued for transmission or transmitted at least once shall not be changed.

Note: Because Link Layer PDUs are not required to be processed in real time, it is possible for the local Controller to have queued but not yet transmitted an LL_LENGTH_REQ PDU when it receives an LL_LENGTH_REQ PDU from the peer device. In this situation each device responds as normal; the resulting collision is harmless.

Upon receiving an LL_LENGTH_REQ PDU, the Link Layer shall respond with an LL_LENGTH_RSP PDU containing its own *connMaxTxOctets*, *connMaxRxOctets*, *connMaxTxTime*, and *connMaxRxTime* values for the connection (which it may have updated based on the values received, for example so as to allow the remote device to transmit longer packets).

If the peer device does not support the LE Coded PHY feature, then the MaxRxTime and MaxTxTime fields in the LL_LENGTH_REQ and LL_LENGTH_RSP PDUs shall be set to a value less than or equal to 2120 microseconds.

If the Link Layer of the master or slave sends the LL_LENGTH_REQ PDU to a device that does not understand that PDU, then the device should expect an LL_UNKNOWN_RSP PDU in response.

The procedure is completed when the initiating Controller receives either an LL_LENGTH_RSP PDU or, in the case the remote device does not support the LE Data Packet Length Extension feature, an LL_UNKNOWN_RSP PDU with the Unknown type set to LL_LENGTH_REQ.

5.1.10 PHY Update Procedure

The PHY Update Procedure, when supported, is used to change the transmit or receive PHYs, or both. The procedure can be initiated either on a request by the Host or autonomously by the Link Layer. Either the master or the slave may initiate this procedure at any time after entering the Connection State. Link Layer PHY preferences may change during a connection or between connections and, therefore, they should not be cached by the peer device.

When this procedure is initiated by the master, it sends an LL_PHY_REQ PDU. The slave responds with an LL_PHY_RSP PDU. The master then responds to this with an LL_PHY_UPDATE_IND PDU.

When this procedure is initiated by the slave, it sends an LL_PHY_REQ PDU. The master responds with an LL_PHY_UPDATE_IND PDU.

The TX_PHYS and RX_PHYS fields of the LL_PHY_REQ and LL_PHY_RSP PDUs shall be used to indicate the PHYs that the sending Link Layer prefers to use. If the sender wants a symmetric connection (one where the two PHYs are the same) it should make both fields the same, only specifying a single PHY.

The M_TO_S_PHY and S_TO_M_PHY fields of the LL_PHY_UPDATE_IND PDU shall indicate the PHYs that shall be used after the instant.

If the master initiated the procedure, it shall determine the PHY to use in each direction based on the contents of the LL_PHY_REQ and LL_PHY_RSP PDUs using the following rules:

- the M_TO_S_PHY field of the LL_PHY_UPDATE_IND PDU shall be determined from the master's TX_PHYS field and the slave's RX_PHYS field;

- the S_TO_M_PHY field of the LL_PHY_UPDATE_IND PDU shall be determined from the master's RX_PHYS field and the slave's TX_PHYS field.

In each of those cases the following rules apply:

- if, for at least one PHY, the corresponding bit is set to 1 in both the TX_PHYS and RX_PHYS fields, the master shall select any one of those PHYs for that direction;
- if there is no PHY for which the corresponding bit is set to 1 in both the TX_PHYS and RX_PHYS fields, the master shall not change the PHY for that direction.

If the slave initiated the procedure, the master shall determine the PHY to use in each direction based on the contents of the LL_PHY_REQ PDU sent by the slave using the following rules:

- the M_TO_S_PHY field of the LL_PHY_UPDATE_IND PDU shall be determined from the RX_PHYS field of the slave’s PDU;
- the S_TO_M_PHY field of the LL_PHY_UPDATE_IND PDU shall be determined from the TX_PHYS field of the slave’s PDU.

In each of those cases the following rules apply:

- if, for at least one PHY, the PHY is one that the master prefers to use and the corresponding bit is set to 1 in the relevant field of the slave’s PDU, the master shall select any one of those PHYs for that direction;
- if there is no PHY which the master prefers to use and for which the corresponding bit is set to 1 in the relevant field of the slave’s PDU, the master shall not change the PHY for that direction.

The remainder of this section shall apply irrespective of which device initiated the procedure.

Irrespective of the above rules, the master may leave both directions unchanged. If the slave specified a single PHY in both the TX_PHYS and RX_PHYS fields and both fields are the same, the master shall either select the PHY specified by the slave for both directions or shall leave both directions unchanged.

If either PHY will change, Section 5.5 shall apply to the LL_PHY_UPDATE_IND PDU. Both devices shall use the new PHYs starting at the instant.

If a master or slave sends an LL_PHY_REQ PDU to a device that does not understand that PDU, then the receiving device shall send an LL_UNKNOWN_RSP PDU in response.

The procedure has completed when:

- an LL_UNKNOWN_RSP or LL_REJECT_EXT_IND PDU has been sent or received;
- an LL_PHY_UPDATE_IND PDU indicating that neither PHY will change has been sent or received; or
- the master sends an LL_PHY_UPDATE_IND PDU indicating that at least one PHY will change and the instant has been reached. In this case, the procedure response timeout shall be stopped on the master when it sends that PDU and on the slave when it receives that PDU.

The Controller shall notify the Host of the PHYs now in effect when the PHY Update Procedure completes if either it has resulted in a change of one or both PHYs or if the procedure was initiated by a request from the Host. Otherwise, it shall not notify the Host that the procedure took place.

5.1.10.1 Packet Transmit Time Restrictions

After having sent or received an LL_PHY_UPDATE_IND PDU that changes either PHY, and until the instant has been reached, the Link Layer shall not transmit a packet that would take longer than *connEffectiveMaxTxTime* microseconds (see Section 4.5.10) to transmit on the PHY that will apply after the instant.

After a slave sends an LL_PHY_REQ PDU, and until it receives an LL_PHY_UPDATE_IND, LL_UNKNOWN_RSP, or LL_REJECT_EXT_IND PDU in response, its link layer shall not transmit a packet that would take longer than *connEffectiveMaxTxTime* microseconds to transmit on any PHY that appears in the TX_PHYS field of that LL_PHY_REQ PDU.

If a slave responds to the PHY Update Procedure then, during the period starting when it sends the LL_PHY_RSP PDU and ending when it receives the LL_PHY_UPDATE_IND PDU in response, the slave's Link Layer shall not transmit a packet that would take longer than *connEffectiveMaxTxTime* microseconds to transmit on any PHY that appears in both the TX_PHYS field of its LL_PHY_RSP PDU and the RX_PHYS field of the master's LL_PHY_REQ PDU.

If the value of *connEffectiveMaxTxTime* changes during the procedure, the above requirements apply to the value at the moment the Data Channel PDU is queued for transmission.

Note: The requirements of this section are in addition to, and do not override, those in Section 4.5.10.

Note: If a Link Layer has any Data Channel PDUs queued for transmission at the start of the procedure or queues any during the procedure, it may need to re-fragment those PDUs in order to obey these requirements.

5.1.11 Minimum Number Of Used Channels Procedure

A Controller uses the Minimum Number Of Used Channels Procedure to request that the peer device uses a minimum number of channels on a given PHY.

The slave can initiate this procedure at any time after entering the Connection State by sending an LL_MIN_USED_CHANNELS_IND PDU. The Master shall not send this PDU.

If the Link Layer receives an LL_MIN_USED_CHANNELS_IND PDU, it should ensure that, whenever the slave-to-master PHY is one of those specified, the connection uses at least the number of channels given in the MinUsedChannels field of the PDU.

The procedure has completed when the Link Layer acknowledgment of the LL_MIN_USED_CHANNELS_IND PDU is sent or received.

If the channel map does not include the minimum number of channels the slave requires for regulatory compliance, the slave must take steps to remain regulatory compliant, which can include disconnecting the link or reducing the output power.

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.

5.3 PROCEDURE COLLISIONS

Since LL Control PDUs are not interpreted in real time, collisions can occur where the Link Layer of the master and the Link Layer of the slave initiate incompatible procedures. Two procedures are incompatible if they both involve an instant. In this situation, the rules in this section shall be followed:

A device shall not initiate a procedure after responding to a PDU that had initiated an incompatible procedure until that procedure is complete.

If device initiates a procedure A and, while that procedure is not complete, receives a PDU from its peer that initiates an incompatible procedure B, then:

- If the peer has already sent at least one PDU as part of procedure A, the device should immediately exit the Connection State and transition to the Standby State.
- Otherwise, if the device is the master, it shall reject the PDU received from the slave by issuing an LL_REJECT_EXT_IND (if supported by both devices) or LL_REJECT_IND (otherwise) PDU. It shall then proceed with procedure A.
- Otherwise (the device is the slave) it shall proceed to handle the master-initiated procedure B and take no further action in the slave-initiated procedure A except processing the rejection from the master.

The Host shall be notified that the link has been disconnected with, or the rejection PDU shall use (as appropriate):

- the error code *LMP Error Transaction Collision / LL Procedure Collision* (0x23) if procedures A and B are the same procedure;
- the error code *LMP Error Transaction Collision / LL Procedure Collision* (0x23) if procedure A is the Connection Update Procedure and procedure B is the Connection Parameters Update Procedure;
- the error code *Different Transaction Collision* (0x2A) otherwise.

5.4 LE AUTHENTICATED PAYLOAD TIMEOUT

LE Authenticated Payload Timeout (*authenticatedPayloadTO*) is a parameter that defines the maximum amount of time in milliseconds allowed between receiving packets containing a valid MIC. The Host can change the value of *authenticatedPayloadTO* using the HCI_Write_Authenticated_Payload_Timeout Command ([Vol 2] Part E, Section 7.3.94). The default value for *authenticatedPayloadTO* is 30 seconds.

When the connection is encrypted, a device supporting LE Ping feature shall start the LE Authenticated Payload timer $T_{LE\_Authenticated\_Payload}$ to monitor the time since the last reception of a packet containing a valid MIC from the remote device. Each device shall reset the timer $T_{LE\_Authenticated\_Payload}$ upon reception of a packet with a valid MIC.

If at any time in the CONNECTION state the timer $T_{LE\_Authenticated\_Payload}$ reaches the *authenticatedPayloadTO* value, the Host shall be notified using the HCI Authenticated Payload Timeout Expired event ([Vol 2] Part E, Section 7.7.75). The $T_{LE\_Authenticated\_Payload}$ Timer restarts after it is expired.

The timer $T_{LE\_Authenticated\_Payload}$ shall continue to run during encryption pause procedure.

Whenever the Host sets the *authenticatedPayloadTO* while the timer $T_{LE\_Authenticated\_Payload}$ is running, the timer shall be reset.

5.5 PROCEDURES WITH INSTANTS

Where a procedure involves a PDU with an Instant field, then the following rules shall apply.

The Instant field shall be used to indicate the *connEventCount* when the relevant change shall be applied; this is known as the instant for the procedure. The master should allow a minimum of 6 connection events that the slave will be listening for before the instant occurs, considering that the slave may only be listening once every *connSlaveLatency* events.

When a slave receives such a 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 acknowledgment of the PDU or *connEventCount* equals Instant.

When a slave receives such a 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, shall exit the Connection State and transition to the Standby State, and shall notify the Host using the error code *Instant Passed* (0x28).

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.

6 PRIVACY

The Link Layer provides Privacy by using Private Addresses (see Section 1.3.2).

If a device is using Resolvable Private Addresses Section 1.3.2.2, it shall also have an Identity Address that is either a Public or Random Static address type.

6.1 PRIVATE ADDRESS GENERATION INTERVAL

A private address shall be generated using the Resolvable Private Address Generation (see Section 1.3.2.2).

The Link Layer shall set a timer determined by the Host. A new private address shall be generated when the timer expires. If the Link Layer is reset, a new private address shall be generated and the timer started with any value in the allowed range.

Note: If the private address is generated frequently, connection establishment times may be affected. It is recommended to set the timer to 15 minutes.

6.2 PRIVACY IN THE ADVERTISING STATE

Privacy in the advertising state determines how the Link Layer processes Resolvable Private Addresses for advertising events.

The requirements in the following sub-sections apply in addition to those in Section 4.4.2.

6.2.1 Connectable and Scannable Undirected Event Type

The Link Layer may use resolvable private addresses or non-resolvable private addresses for the advertiser’s device address (AdvA field) when entering the Advertising State and using connectable and scannable undirected events.

The AdvA field of the connectable and scannable undirected advertising event PDU is generated using the Local IRK value and the Resolvable Private Address Generation Procedure (see Section 1.3.2.2). If the Host has not provided any Resolving List IRK pairs for the peer to the Link Layer, then the AdvA field shall use a Host-provided address.

When an advertiser receives a connection request that contains a resolvable private address for the initiator’s address (InitA field), the Link Layer shall resolve the private address (see Section 1.3.2.3). The advertising filter policy, where the White List is enabled (see Section 4.3.2), shall determine if the advertiser establishes a connection.

When an advertiser receives a connection request that contains a device identity address for the initiator's address field (InitA field), and if that device is in the Resolving List with a non-zero peer IRK for which the Host has specified device privacy mode, then the advertising filter policy, where the White List is enabled (see Section 4.3.2), shall determine if the advertiser establishes a connection.

When an advertiser receives a scan request that contains a resolvable private address for the scanner’s device address (ScanA field), the Link Layer shall resolve the private address (see Section 1.3.2.3). The advertising filter policy, where the White List is enabled (see Section 4.3.2), shall determine if the advertiser processes the scan request.

When an advertiser receives a scan request that contains a device identity address for the scanner's device address (ScanA field), and if that device is in the Resolving List with a non-zero peer IRK for which the Host has specified device privacy mode, then the advertising filter policy, where the White List is enabled (see Section 4.3.2), shall determine if the advertiser processes the scan request.

When an advertiser receives a scan or connection request that contains a non-resolvable private address, the advertising filter policy (see Section 4.3.2) shall determine if the advertiser processes the scan or connection request.

If the advertiser processes the scan request, the advertiser’s device address (AdvA field) in the SCAN_RSP PDU shall be the same as the advertiser’s device address (AdvA field) in the SCAN_REQ PDU to which it is responding.

6.2.2 Connectable Directed Event Type

The Link Layer shall use resolvable private addresses for the advertiser’s device address (AdvA field). If an IRK is available in the Link Layer Resolving List for the peer device, then the target’s device address (TargetA field) shall use a resolvable private address. If an IRK is not available in the Link Layer Resolving List or the IRK is set to zero for the peer device, then the target’s device address (TargetA field) shall use the Identity Address when entering the Advertising State and using connectable directed events.

The AdvA field of the connectable directed advertising event PDU is generated using the Local IRK value and the Resolvable Private Address Generation Procedure (see Section 1.3.2.2).

The TargetA field of the connectable directed advertising event PDU is generated using the Peer IRK value and the Resolvable Private Address Generation Procedure (see Section 1.3.2.2). The TargetA field uses the public or static device address of the peer device if no peer IRK is available.

When an advertiser receives a connection request that contains a resolvable private address for the initiator's address (InitA field), the Link Layer shall resolve the private address (see Section 1.3.2.3).

When an advertiser receives a connection request that contains a device identity address for the initiator's address field (InitA field), and if that device is in the Resolving List with a non-zero peer IRK for which the Host has specified device privacy mode, then the advertising filter policy, where the White List is enabled (see Section 4.3.2), shall determine if the advertiser establishes a connection.

6.2.3 Non-connectable and Non-scannable Undirected and Scannable Undirected Event Types

The Link Layer may use resolvable private addresses or non-resolvable private addresses for the advertiser's device address (AdvA field) when entering the Advertising State and using the following event types:

- non-connectable and non-scannable undirected event
- scannable undirected event

The AdvA field of the non-connectable and non-scannable undirected advertising event PDU and scannable undirected event PDU are generated using the advertiser's Local IRK value and the Resolvable Private Address Generation Procedure or Non-Resolvable Private Address Generation Procedure (see Section 1.3.2.2).

When an advertiser receives a scan request that contains a resolvable private address for the scanner's device address (ScanA field), the Link Layer shall resolve the private address (see Section 1.3.2.3). The advertising filter policy, where the White List is enabled (see Section 4.3.2), shall determine if the advertiser processes the scan request.

When an advertiser receives a scan request that contains a device identity address for the scanner's device address (ScanA field), and if that device is in the Resolving List with a non-zero peer IRK for which the Host has specified device privacy mode, then the advertising filter policy, where the White List is enabled (see Section 4.3.2), shall determine if the advertiser processes the scan request.

When an advertiser receives a scan request that contains a non-resolvable private address, the advertising filter policy (see Section 4.3.2) shall determine if the advertiser processes the scan request.

If the advertiser processes the scan request, the advertiser's device address (AdvA field) in the scan response PDU shall be the same as the advertiser's device address (AdvA field) in the scan request PDU to which it is responding.

6.2.4 Connectable Undirected Event Type

The Link Layer may use resolvable private addresses or non-resolvable private addresses for the advertiser's device address (AdvA field) when entering the Advertising State and using connectable undirected events.

The AdvA field of the connectable undirected advertising event PDU is generated using the Local IRK value and the Resolvable Private Address Generation Procedure (see Section 1.3.2.2). If the Host has not provided any Resolving List IRK pairs for the peer to the Link Layer, then the AdvA field shall use a Host-provided address.

When an advertiser receives a connection request that contains a resolvable private address for the target's address (TargetA field), the Link Layer shall resolve the private address (see Section 1.3.2.3). The advertising filter policy, where the White List is enabled (see Section 4.3.2), shall determine if the advertiser establishes a connection.

The advertising filter policy (see Section 4.3.2) shall determine if the advertiser processes the connect request if an advertiser receives a connect request that contains a non-resolvable private address.

6.2.5 Non-connectable and Non-scannable Directed and Scannable Directed Event Types

The Link Layer may use resolvable private addresses or non-resolvable private addresses for the advertiser's device address (AdvA field) when entering the Advertising State and using the following event types:

- non-connectable and non-scannable directed event
- scannable directed event

If an IRK is available in the Link Layer Resolving List for the peer device, then the target's device address (TargetA field) shall use a resolvable private address. If an IRK is not available in the Link Layer Resolving List or the IRK is set to zero for the peer device, then the target's device address (TargetA field) shall use the Identity Address when entering the Advertising State and using non-connectable and non-scannable directed and scannable directed events.

The AdvA field of the non-connectable and non-scannable directed advertising event PDU and scannable directed event PDU is generated using the advertiser's Local IRK value and the Resolvable Private Address Generation Procedure or Non-Resolvable Private Address Generation Procedure (see Section 1.3.2.2).

The TargetA field of the scannable directed advertising event PDU is generated using the Peer IRK value and the Resolvable Private Address Generation Procedure (see Section 1.3.2.2). The TargetA field uses the public or static device address of the peer device if no peer IRK is available.

When an advertiser receives a scan request that contains a resolvable private address for the scanner's device address (ScanA field), the Link Layer shall resolve the private address (see Section 1.3.2.3).

If the advertiser processes the scan request, the advertiser's device address (AdvA field) in the AUX_SCAN_RSP PDU shall be the same as the advertiser's device address (AdvA field) in the AUX_SCAN_REQ PDU to which it is responding.

6.3 PRIVACY IN THE SCANNING STATE

The requirements in this section apply in addition to those in Section 4.4.3.

The Link Layer may use resolvable private addresses or non-resolvable private addresses for the scanner's device address (ScanA field) when entering the Scanning State.

The ScanA field of the scanning PDU is generated using the Resolving List's Local IRK value and the Resolvable Private Address Generation Procedure (see Section 1.3.2.2), or the address is provided by the Host.

The advertiser's device address (AdvA field) in the scan request PDU shall be the same as the advertiser's device address (AdvA field) received in the advertising PDU to which the scanner is responding.

When a scanner receives an advertising event that contains a resolvable private address for the advertiser's device address (AdvA field), the Link Layer shall resolve the private address (see Section 1.3.2.3). The scanner's filter policy, where the White List is enabled (see Section 4.3.3), shall determine if the scanner responds with a scan request.

When a scanner receives an advertising packet that contains a device identity address for the advertiser's device address (AdvA field), and if that device is in the Resolving List with a non-zero peer IRK for which the Host has specified device privacy mode, then the scanner's filter policy, where the White List is enabled (see Section 4.3.3), shall determine if the scanner responds with a scan request.

When a scanner receives an advertising event that contains a non-resolvable private address, the scanner's filter policy (see Section 4.3.3) shall determine if the scanner processes the advertising event.

6.4 PRIVACY IN THE INITIATING STATE

The requirements in this section apply in addition to those in Section 4.4.4.

The Link Layer may use resolvable private addresses or an address provided by the Host for the initiator’s device address (InitA field) when in the Initiating state.

When an initiator receives a connectable advertising event that contains a resolvable private address for the advertiser’s address (AdvA field), the Link Layer shall resolve the private address using the Peer IRK values (see Section 1.3.2.3). The initiator’s filter policy (see Section 4.3.4) shall determine if the initiator establishes a connection.

The advertiser’s device address (AdvA field) in the initiating PDU shall be the same as the advertiser’s device address (AdvA field) received in the advertising event PDU to which the initiator is responding.

When an initiator receives a directed connectable advertising event that contains a resolvable private address for the target’s address (TargetA field), the Link Layer shall resolve the private address using the Local IRK values (see Section 1.3.2.3). An initiator that has been instructed by the Host to use Resolvable Private Addresses shall not respond to directed connectable advertising events that contain Public or Static addresses for the target’s address (TargetA field).

When an initiator receives a connectable advertising event that contains a device identity address for the advertiser's device address (AdvA field), and if that device is in the Resolving List with a non-zero peer IRK for which the Host has specified device privacy mode, then the initiator's filter policy, where the White List is enabled (see Section 4.3.4), shall determine if the initiator establishes a connection.

The Link Layer shall use resolvable private addresses for the initiator’s device address (InitA field) when initiating connection establishment with an associated device that exists in the Resolving List. The initiator’s device address (InitA field) in the initiating PDU is generated using the Resolving List Local IRK and the Resolvable Private Address Generation Procedure (see Section 1.3.2.2). The Link Layer should not set the InitA field to the same value as the TargetA field in the received advertising PDU.

The Link Layer shall use the Host-provided address for the initiator’s device address (InitA field) when initiating connection establishment with a device that is not in the Resolving List.

6.5 PRIVACY OF THE DEVICE

A private device shall not use its Identity Address in any advertising PDU. The Host may command the Controller to advertise, scan, or initiate a connection using a Resolvable Private Address when the resolving list is enabled. If the local IRK in the resolving list associated with the peer Identity Address is all zeros, the Controller will use the Identity Address. If the peer IRK in the resolving list associated with the peer Identity Address is all zeros, the Controller will accept the Identity Address. If the Host has instructed the Controller to use device privacy mode with a peer Identity Address, the Controller will accept the peer’s Identity Address. This implies that the device's network privacy is violated. To maintain a device’s network privacy, the Host should only populate entries in the Controller’s resolving list with non-zero IRKs and not instruct the Controller to use device privacy mode.

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) 0x0800
    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 0x01
  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) 0x0800
    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) 0x0801
    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 the notation: LSbyte to MSbyte & 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
```

2 LE CODED PHY SAMPLE DATA

Whenever bits are specified, they are in transmission order irrespective of spacing.

2.1 REFERENCE INFORMATION PACKET

The reference packet is described as bytes in transmission order (the leftmost byte in a line is transmitted first). Inside a byte, bits are transmitted LSB first.

```
Access address:   D6 BE 89 8E
PDU:              00 03 42 4C 45
CRC:              29 0A CE
```

2.2 FORWARD ERROR CORRECTION ENCODER

This data shows the bits input to and output by the FEC encoder and its internal state.

The encoder state is expressed in octal notation where the LSB represents the rightmost bit store in Figure 3.5 of [Vol 6] Part B, Section 3.3.1. The state specified is that after the bits are output and the shift operations have taken place.

```
Access address
Input:    0   1   1   0   1   0   1   1   0   1   1   1   1   1   0   1
          1   0   0   1   0   0   0   1   0   1   1   1   0   0   0   1
State:    0   4   6   3   5   2   5   6   3   5   6   7   7   7   3   5
          6   3   1   4   2   1   0   4   2   5   6   7   3   1   0   4
Output:   0 0 1 1 0 1 0 1 1 1 0 1 0 0 1 0 0 1 1 1 1 0 1 0 0 1 0 1 1 0 1 1
          1 0 0 1 0 0 0 0 1 0 1 1 1 1 1 1 1 0 0 0 1 0 1 0 1 0 0 0 1 1 1 1
```

Note to reviewers: this spacing makes it more obvious where the bits come from.

```
CI

          If S=2                If S=8
Input:    1   0                 0   0
State:    6   3                 2   1
Output:   0 1 0 1               1 0 1 1

TERM1
          If S=2                If S=8
Input:    0   0   0             0   0   0
State:    1   0   0             0   0   0
Output:   0 0 1 1 0 0           1 1 0 0 0 0

PDU

Input:    0   0   0   0   0   0   0   0   1   1   0   0   0   0   0   0
          0   1   0   0   0   0   1   0   0   0   1   1   0   0   1   0
          1   0   1   0   0   0   1   0
State:    0   0   0   0   0   0   0   0   4   6   3   1   0   0   0   0
          0   4   2   1   0   0   4   2   1   0   4   6   3   1   4   2
```

```
            5   2   5   2   1   0   4   2
Output :    0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 0 1 0 1 0 0 1 1 0 0 0 0 0 0
            0 0 1 1 1 0 1 1 1 1 0 0 1 1 1 0 1 1 1 1 1 1 0 1 0 1 0 0 0 0 1 0
            0 0 0 1 0 0 0 1 1 1 1 1 1 1 1 0
CRC
Input :     1   0   0   1   0   1   0   0   0   1   0   1   0   0   0   0
            0   1   1   1   0   0   1   1
State:      5   2   1   4   2   5   2   1   0   4   2   5   2   1   0   0
            0   4   6   7   3   1   4   6
Output :    0 0 0 1 1 1 0 0 1 0 0 0 0 1 1 1 1 1 1 1 1 0 0 0 0 1 1 1 1 1 0 0
            0 0 1 1 0 1 1 0 1 0 0 0 0 0 0 1
TERM2
Input :     0   0   0
State:      3   1   0
Output :    0 1 0 0 1 1
```

2.3 TRANSMITTED SYMBOLS (S=2)

```
Preamble
0011 1100 0011 1100 0011 1100 0011 1100 0011 1100 0011 1100
0011 1100 0011 1100 0011 1100 0011 1100

Access Address
0011 0011 1100 1100 0011 1100 0011 1100 1100 1100 0011 1100
0011 0011 1100 0011 0011 1100 1100 1100 1100 0011 1100 0011
0011 1100 0011 1100 1100 0011 1100 1100 1100 0011 0011 1100
0011 0011 0011 0011 1100 0011 1100 1100 1100 1100 1100 1100
1100 0011 0011 0011 1100 0011 1100 0011 1100 0011 0011 0011
1100 1100 1100 1100

CI
0011 1100 0011 1100

TERM1
0011 0011 1100 1100 0011 0011

PDU
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 0 1 0 1 0 0 1 1 0 0 0 0 0 0 1 1 1 0 1
1 1 1 0 0 1 1 1 0 1 1 1 1 1 1 0 1 0 1 0 0 0 0 1 0 0 0 0 1 0 0 0 1 1 1 1 1 1 1
1 0

CRC
0 0 0 1 1 1 0 0 1 0 0 0 0 1 1 1 1 1 1 1 1 0 0 0 0 1 1 1 1 1 0 0 0 0 1 1 0 1 1
0 1 0 0 0 0 0 0 1

TERM2
0 1 0 0 1 1
```

Total Packet Duration 510 µs

Note: Groupings of 4 bits show the effects of the P=4 pattern mapper.

2.4 TRANSMITTED SYMBOLS (S=8)

```
Preamble
0011 1100 0011 1100 0011 1100 0011 1100 0011 1100 0011 1100
0011 1100 0011 1100 0011 1100 0011 1100
```

```
Access Address
0011 0011 1100 1100 0011 1100 0011 1100 1100 1100 0011 1100
0011 0011 1100 0011 0011 1100 1100 1100 1100 0011 1100 0011
0011 1100 0011 1100 1100 0011 1100 1100 1100 0011 0011 1100
0011 0011 0011 0011 1100 0011 1100 1100 1100 1100 1100 1100
1100 0011 0011 0011 1100 0011 1100 0011 1100 0011 0011 0011
1100 1100 1100 1100

CI
1100 0011 1100 1100

TERM1
1100 1100 0011 0011 0011 0011

PDU
0011 0011 0011 0011 0011 0011 0011 0011 0011 0011 0011 0011
0011 0011 0011 0011 1100 1100 0011 1100 0011 1100 0011 0011
1100 1100 0011 0011 0011 0011 0011 0011 0011 0011 1100 1100
1100 0011 1100 1100 1100 1100 0011 0011 1100 1100 1100 0011
1100 1100 1100 1100 1100 1100 0011 1100 0011 1100 0011 0011
0011 0011 1100 0011 0011 0011 0011 1100 0011 0011 0011 1100
1100 1100 1100 1100 1100 1100 1100 0011

CRC
0011 0011 0011 1100 1100 1100 0011 0011 1100 0011 0011 0011
0011 1100 1100 1100 1100 1100 1100 1100 1100 0011 0011 0011
0011 1100 1100 1100 1100 1100 0011 0011 0011 0011 1100 1100
0011 1100 1100 0011 1100 0011 0011 0011 0011 0011 0011 1100

TERM2
0011 1100 0011 0011 1100 1100
```

Total Packet Duration 912 µs

Note: Groupings of 4 bits show the effects of the P=4 pattern mapper.

3 LE CHANNEL SELECTION ALGORITHM #2 SAMPLE DATA

This section contains two sets of sample data with different channel maps for the LE Channel Selection Algorithm #2.

The test access address is 0x8E89BED6, meaning the *channelIdentifier* is 0x305F.

3.1 SAMPLE DATA 1 (37 USED CHANNELS)

Channel map [36:0] = 11111_11111111_11111111_11111111_11111111b.

Counter	1	2	3
prn_e	1685	38301	27475
unmappedChannel	20	6	21
mappedChannel	20	6	21

3.2 SAMPLE DATA 2 (9 USED CHANNELS)

Channel map [36:0] =11110_00000000_11100000_00000110_00000000b.

The remapping table is [9, 10, 21, 22, 23, 33, 34, 35, 36].

Counter	6	7	8
prn_e	10975	5490	46970
unmappedChannel	23	14	17
mappedChannel	23	9	34

Core System Package [Low Energy Controller volume]

Part D

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 an 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.

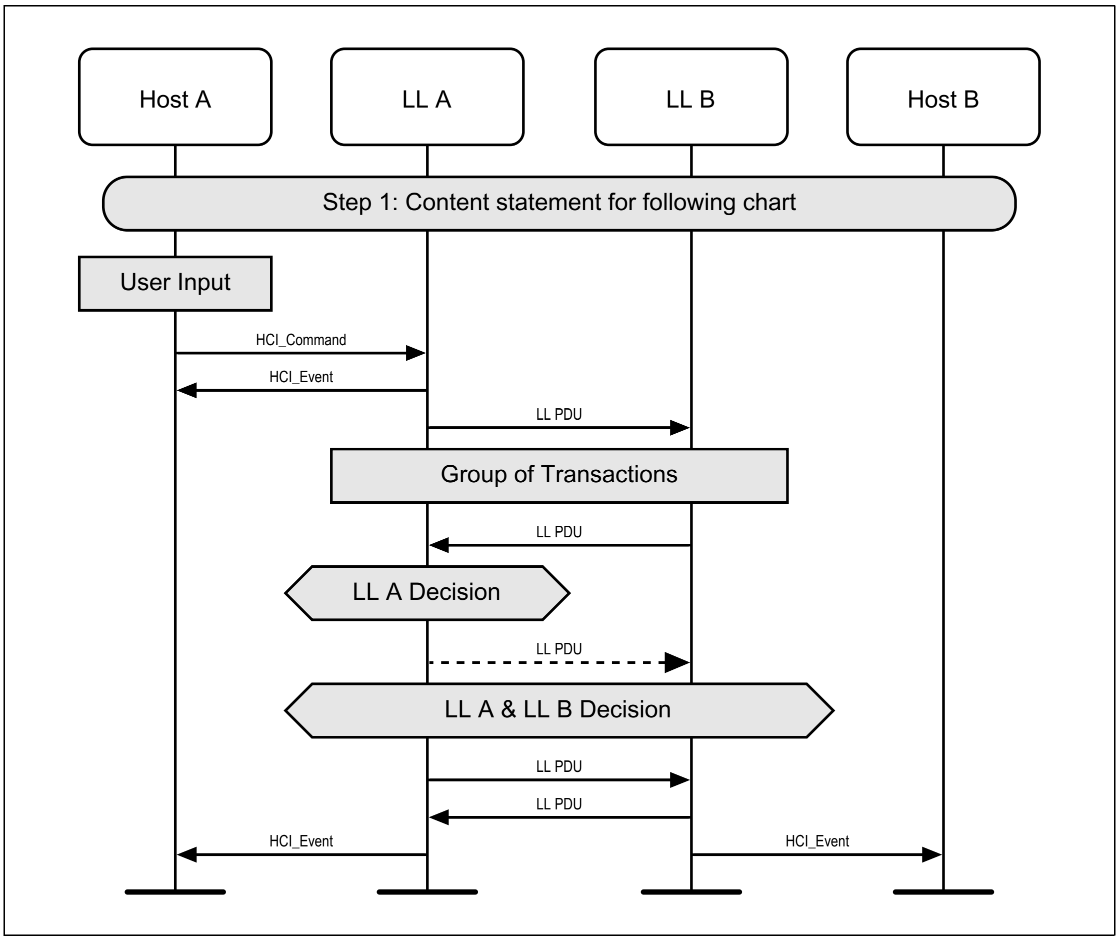


Figure 1.1: Example MSC

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).

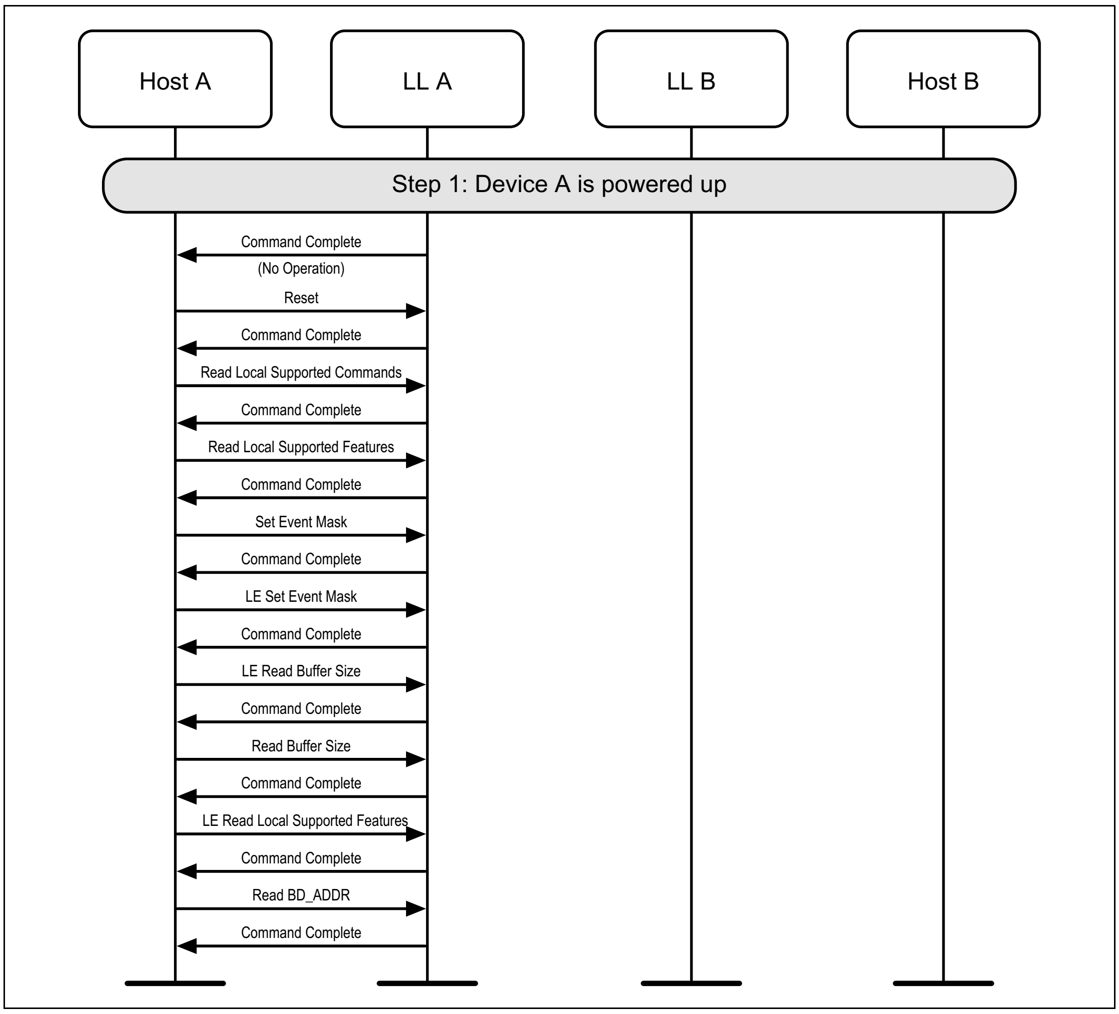


Figure 2.1: Initial Setup

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).

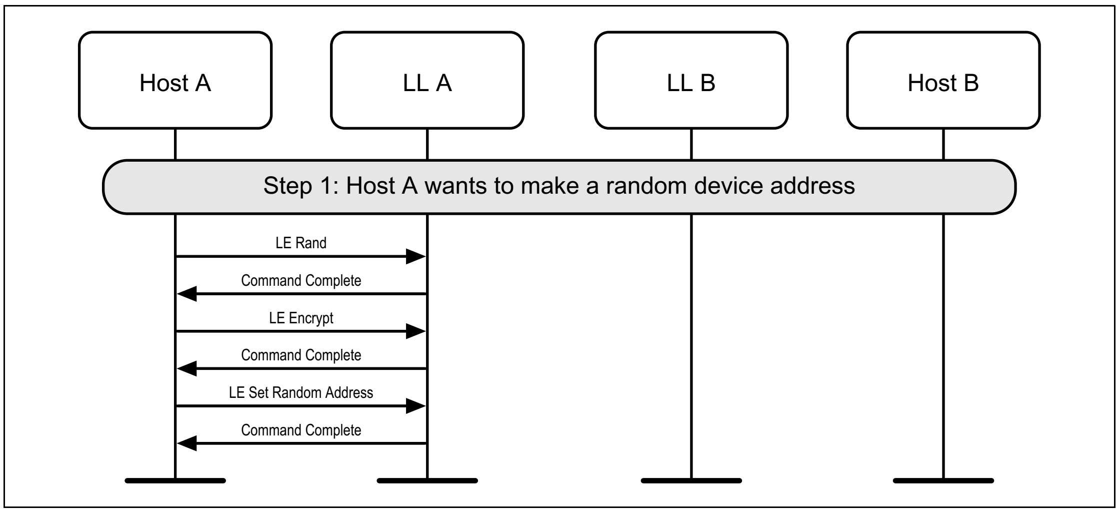


Figure 2.2: Random Device Address

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).

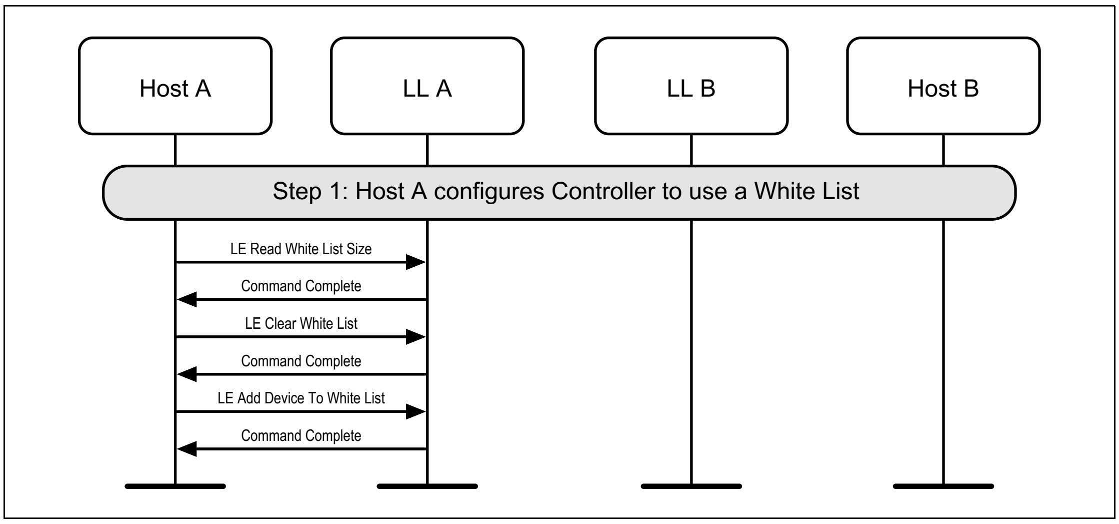


Figure 2.3: White Lists

2.4 ADDING IRK TO RESOLVING LIST

Before advertising, scanning or initiating can use resolving lists, the resolving list may be cleared and devices added in as required (see Figure 2.4).

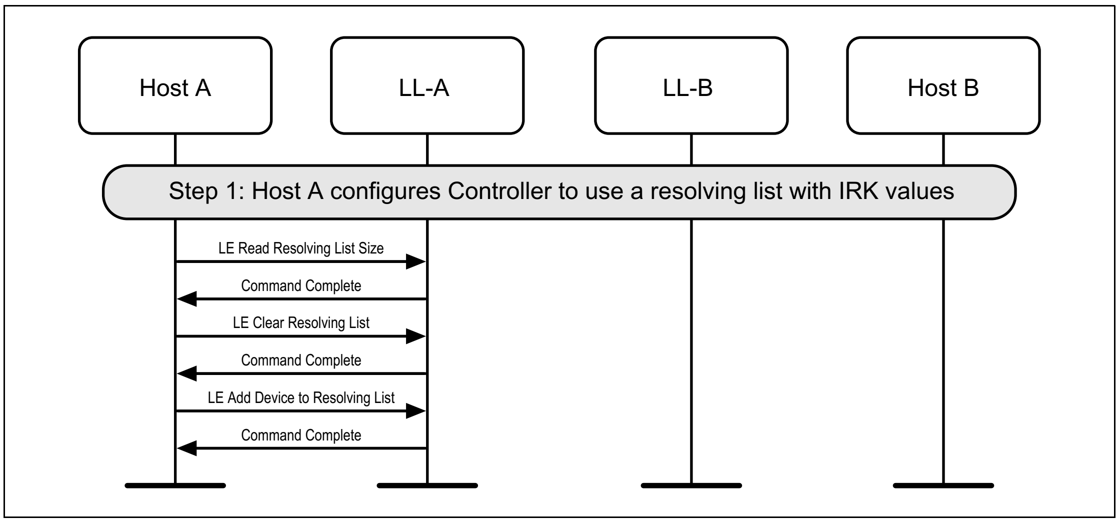

Host A
LL-A
LL-B
Host B
Step 1: Host A configures Controller to use a resolving list with IRK values
LE Read Resolving List Size
Command Complete
LE Clear Resolving List
Command Complete
LE Add Device to Resolving List
Command Complete

Figure 2.4: Resolving Lists

2.5 DEFAULT DATA LENGTH

Before creating a connection, the Host may specify its preferred values for the Controller's maximum transmission packet size and maximum packet transmission time to be used for new connections. This may be done on either the master or the slave.

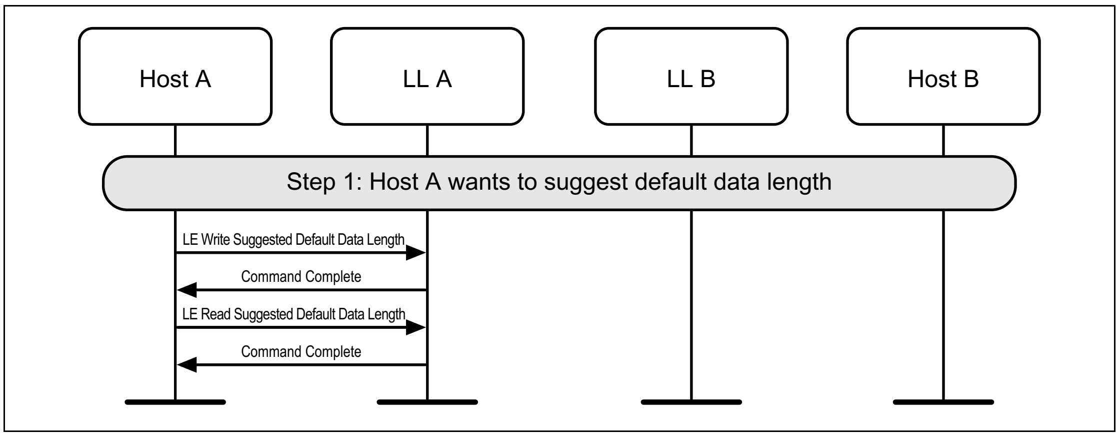

Host A
LL A
LL B
Host B
Step 1: Host A wants to suggest default data length
LE Write Suggested Default Data Length
Command Complete
LE Read Suggested Default Data Length
Command Complete

Figure 2.5: Default Data Length

2.6 PERIODIC ADVERTISER LIST

The Periodic Advertiser List may be cleared and devices added in as required, before it is made use of (see Figure 2.6).

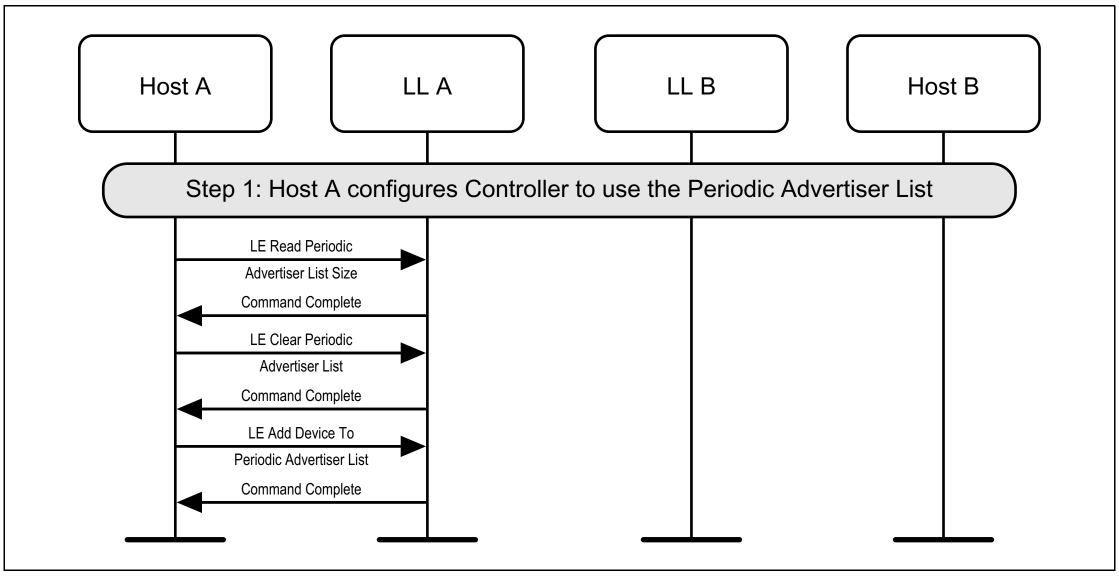

Host A
LL A
LL B
Host B
Step 1: Host A configures Controller to use the Periodic Advertiser List
LE Read Periodic Advertiser List Size
Command Complete
LE Clear Periodic Advertiser List
Command Complete
LE Add Device To Periodic Advertiser List
Command Complete

Figure 2.6: Periodic Advertiser List

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).

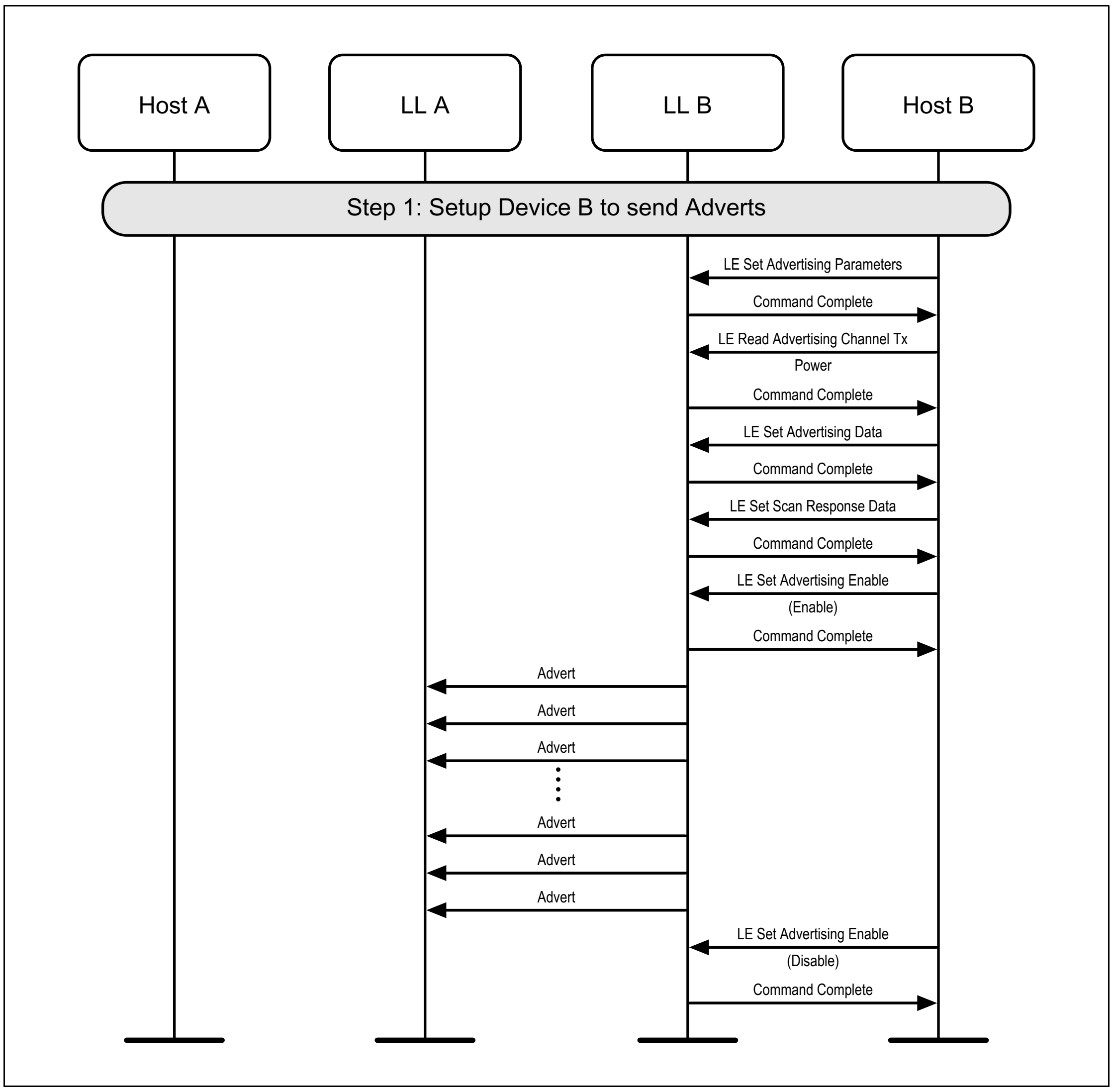

Host A
LL A
LL B
Host B
Step 1: Setup Device B to send Adverts
LE Set Advertising Parameters
Command Complete
LE Read Advertising Channel Tx Power
Command Complete
LE Set Advertising Data
Command Complete
LE Set Scan Response Data
Command Complete
LE Set Advertising Enable (Enable)
Command Complete
Advert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

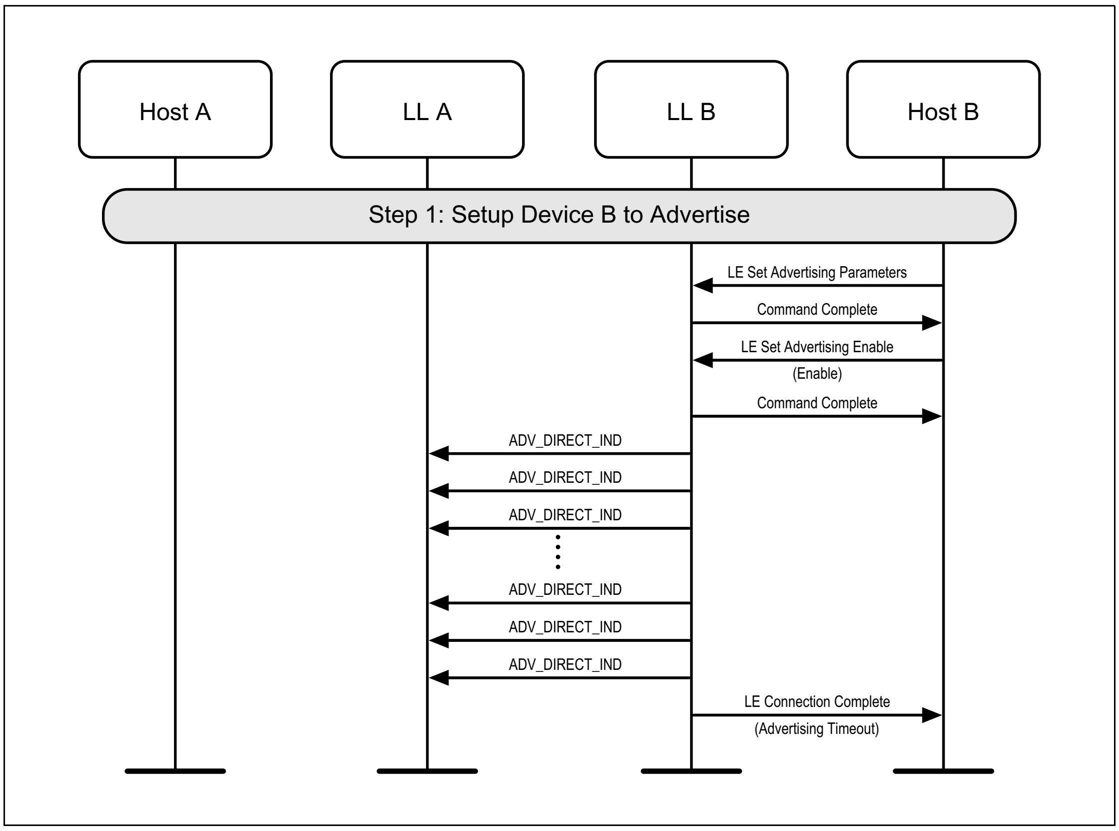


Figure 3.2: High Duty Cycle Directed Advertising showing failure case

Low duty cycle directed advertising must similarly be enabled in order to enter the Advertising State. A device should also configure the advertising parameters before doing this (see Figure 3.3).

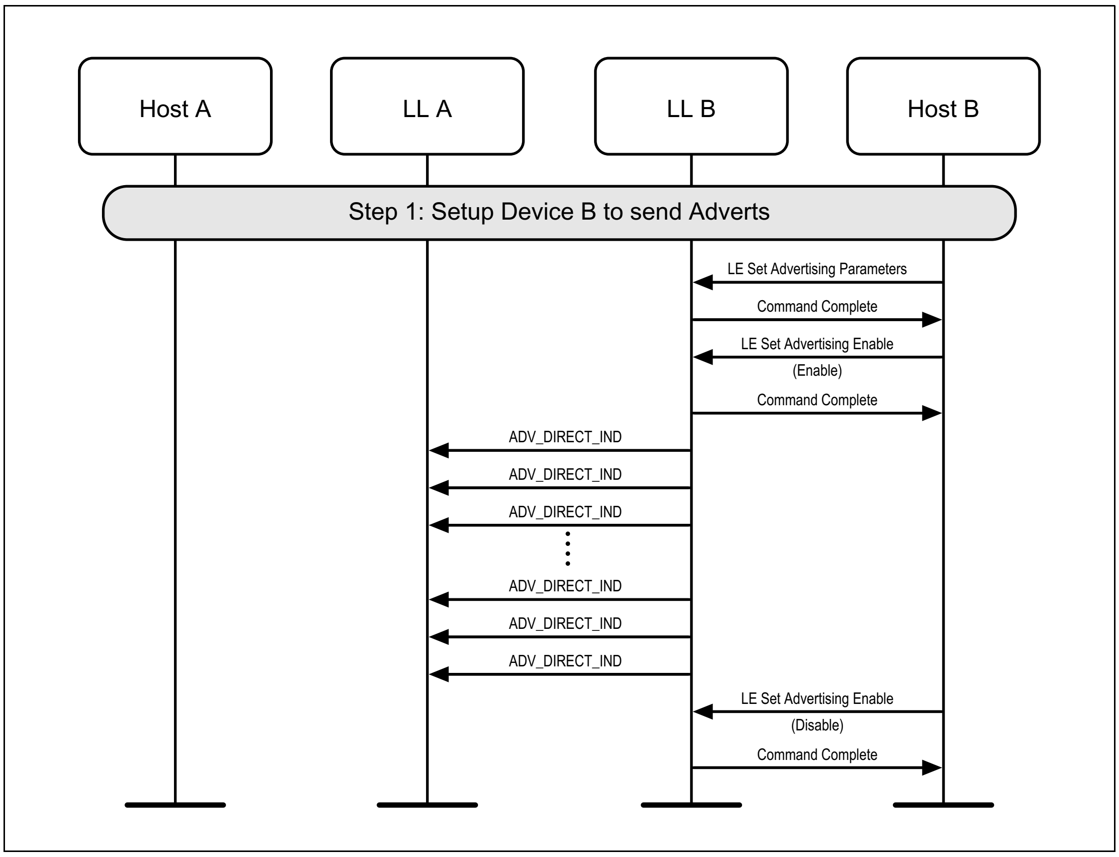


Figure 3.3: Low Duty Cycle Directed Advertising

3.3 ADVERTISING USING ADV_EXT_IND

A device may enter the Advertising State by enabling advertising a set. It should also configure the advertising set parameters before doing this (see Figure 3.4).

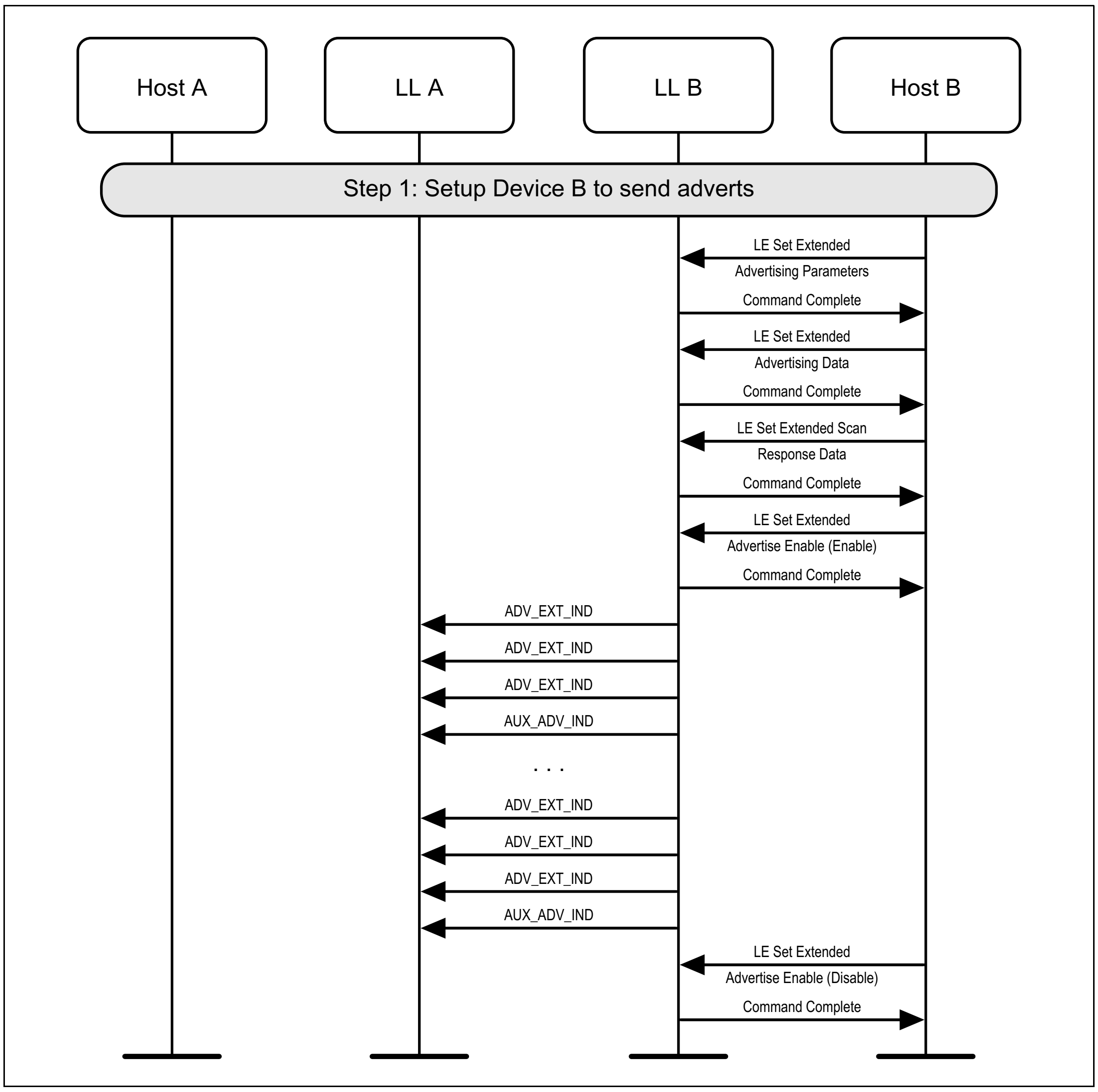

Host A
LL A
LL B
Host B
Step 1: Setup Device B to send adverts
LE Set Extended Advertising Parameters
Command Complete
LE Set Extended Advertising Data
Command Complete
LE Set Extended Scan Response Data
Command Complete
LE Set Extended Advertise Enable (Enable)
Command Complete
ADV_EXT_IND
ADV_EXT_IND
ADV_EXT_IND
AUX_ADV_IND
. . .
ADV_EXT_IND
ADV_EXT_IND
ADV_EXT_IND
AUX_ADV_IND
LE Set Extended Advertise Enable (Disable)
Command Complete

Figure 3.4: Advertising using ADV_EXT_IND

3.4 SCAN REQUEST NOTIFICATIONS

A device may enable scan request notifications in an advertising set (see Figure 3.5).

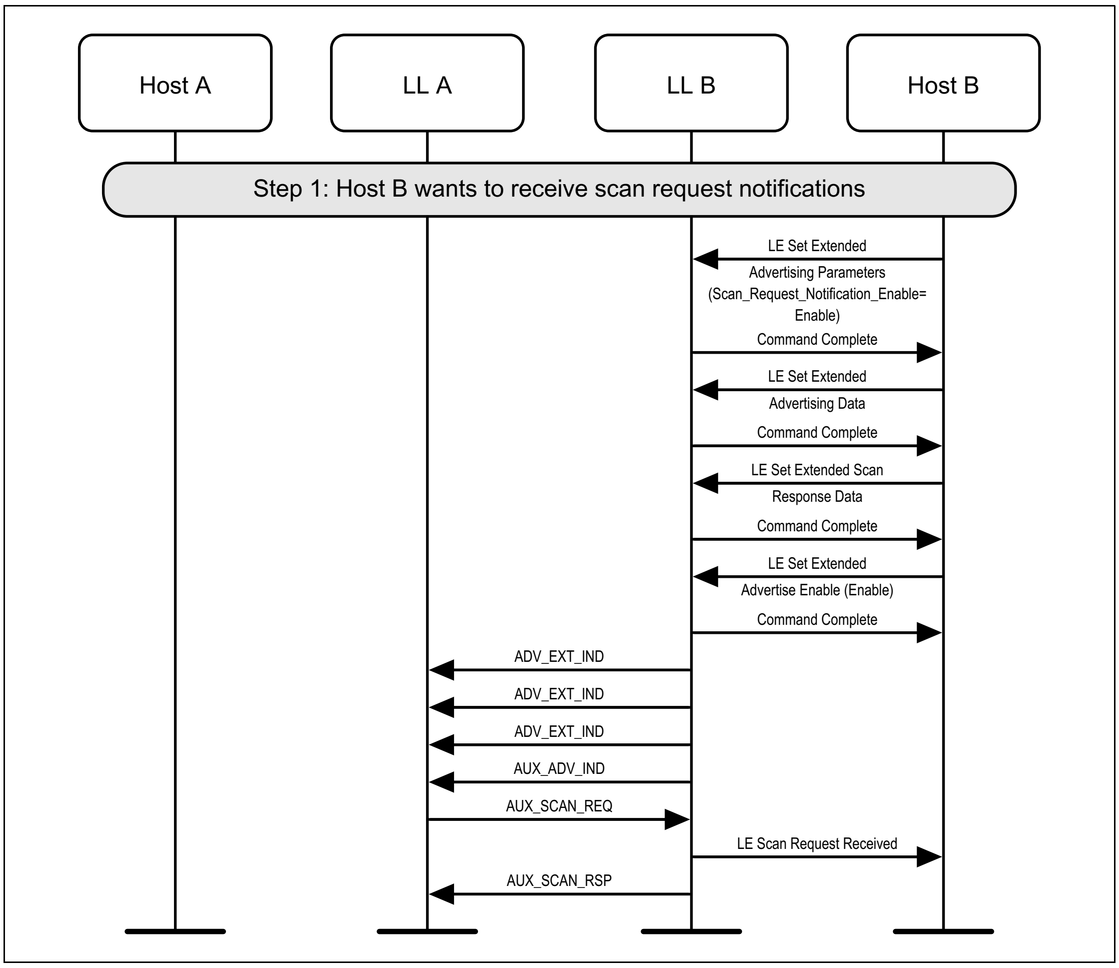

Host A
LL A
LL B
Host B
Step 1: Host B wants to receive scan request notifications
LE Set Extended Advertising Parameters (Scan_Request_Notification_Enable= Enable)
Command Complete
LE Set Extended Advertising Data
Command Complete
LE Set Extended Scan Response Data
Command Complete
LE Set Extended Advertise Enable (Enable)
Command Complete
ADV_EXT_IND
ADV_EXT_IND
ADV_EXT_IND
AUX_ADV_IND
AUX_SCAN_REQ
LE Scan Request Received
AUX_SCAN_RSP

Figure 3.5: Scan Request Notifications

3.5 ADVERTISING DURATION ENDED

A device may enter the Advertising State by enabling advertising a set for a limited duration of time (see Figure 3.6).

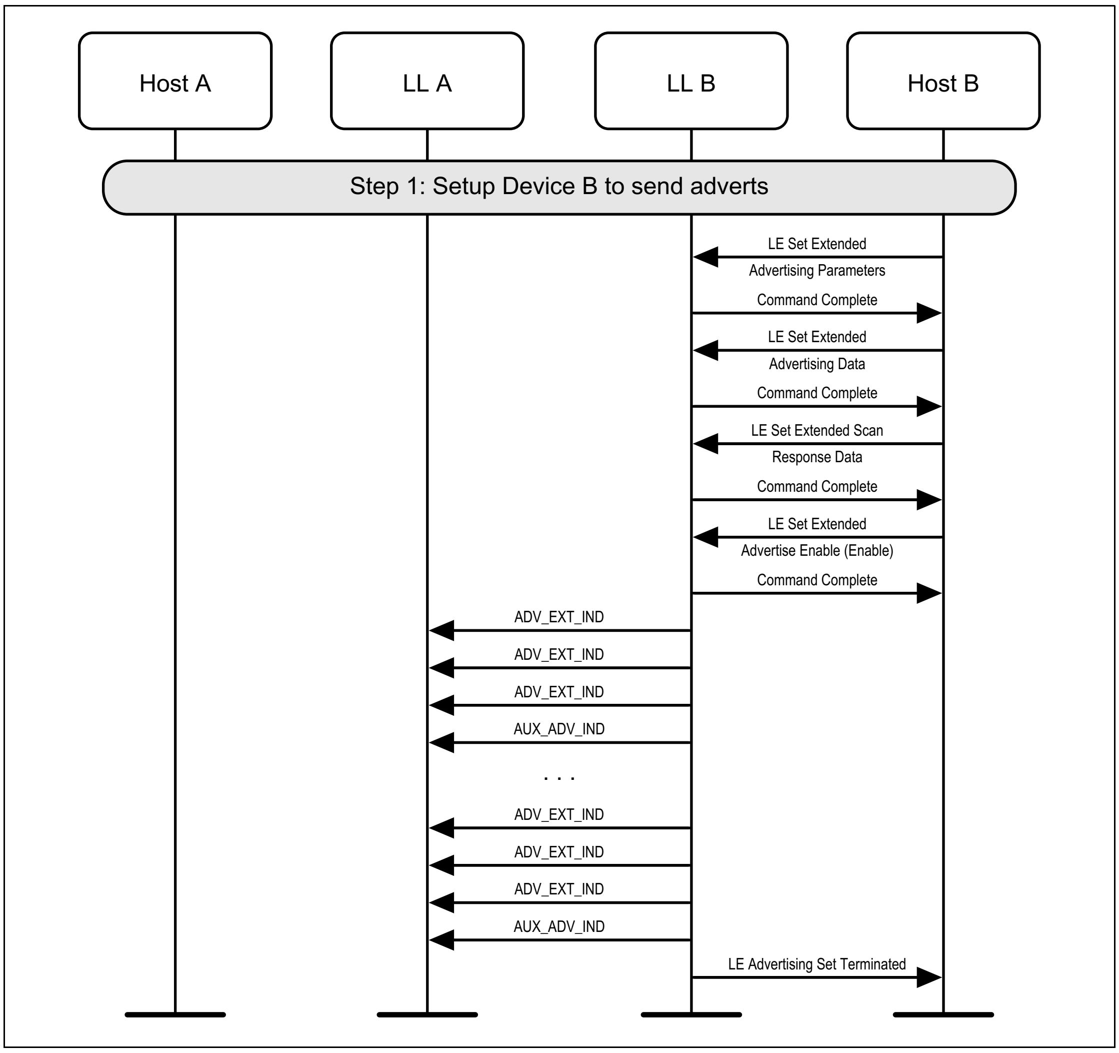

Host A
LL A
LL B
Host B
Step 1: Setup Device B to send adverts
LE Set Extended Advertising Parameters
Command Complete
LE Set Extended Advertising Data
Command Complete
LE Set Extended Scan Response Data
Command Complete
LE Set Extended Advertise Enable (Enable)
Command Complete
ADV_EXT_IND
ADV_EXT_IND
ADV_EXT_IND
AUX_ADV_IND
. . .
ADV_EXT_IND
ADV_EXT_IND
ADV_EXT_IND
AUX_ADV_IND
LE Advertising Set Terminated

Figure 3.6: Advertising Duration Ended

3.6 PERIODIC ADVERTISING

A device may enter the Advertising State by enabling periodic advertising in a set. It should also configure the advertising set parameters before doing this (see Figure 3.7).

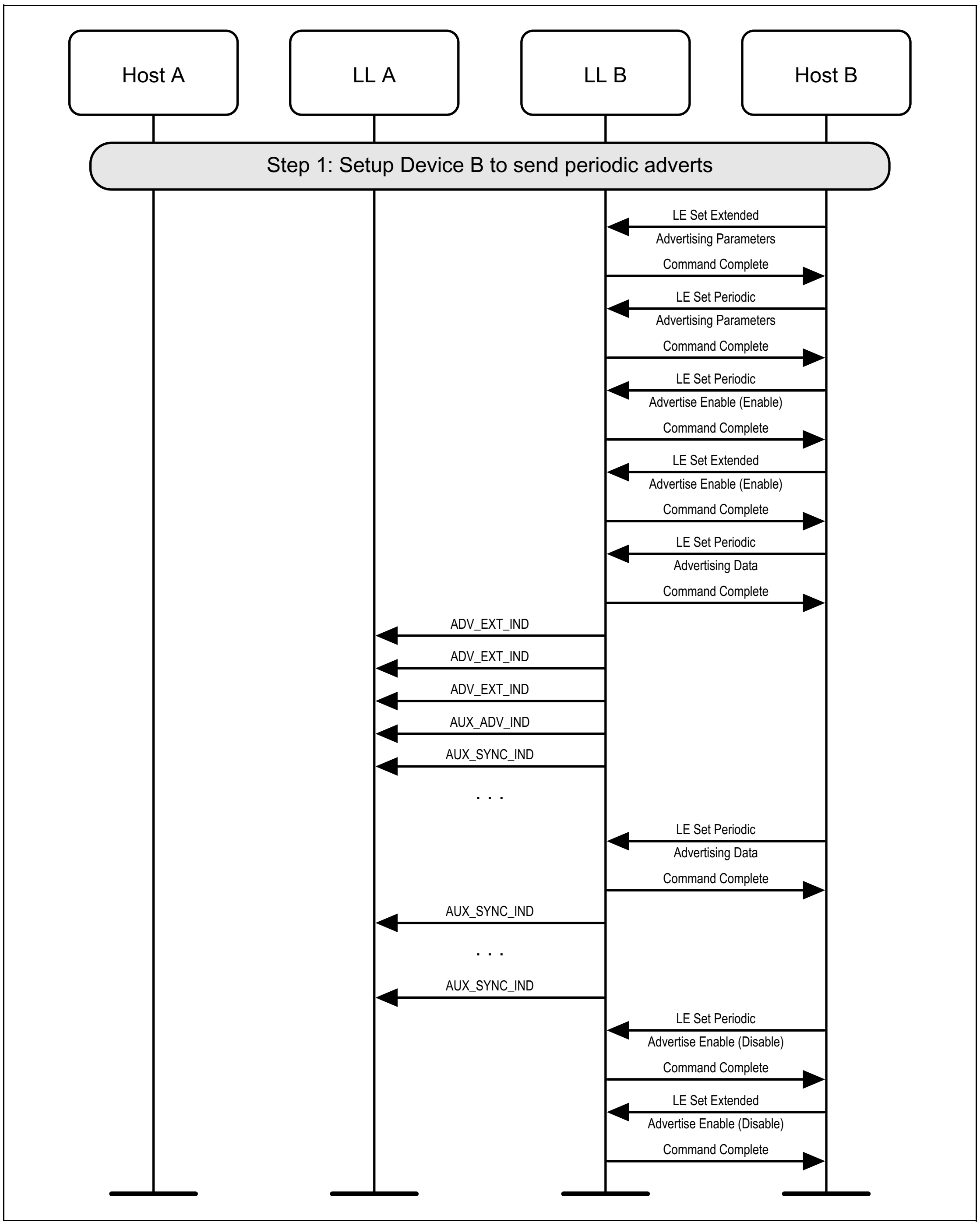

Host A
LL A
LL B
Host B
Step 1: Setup Device B to send periodic adverts
LE Set Extended Advertising Parameters
Command Complete
LE Set Periodic Advertising Parameters
Command Complete
LE Set Periodic Advertise Enable (Enable)
Command Complete
LE Set Extended Advertise Enable (Enable)
Command Complete
LE Set Periodic Advertising Data
Command Complete
ADV_EXT_IND
ADV_EXT_IND
ADV_EXT_IND
AUX_ADV_IND
AUX_SYNC_IND
. . .
LE Set Periodic Advertising Data
Command Complete
AUX_SYNC_IND
. . .
AUX_SYNC_IND
LE Set Periodic Advertise Enable (Disable)
Command Complete
LE Set Extended Advertise Enable (Disable)
Command Complete

Figure 3.7: Periodic Advertising

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).

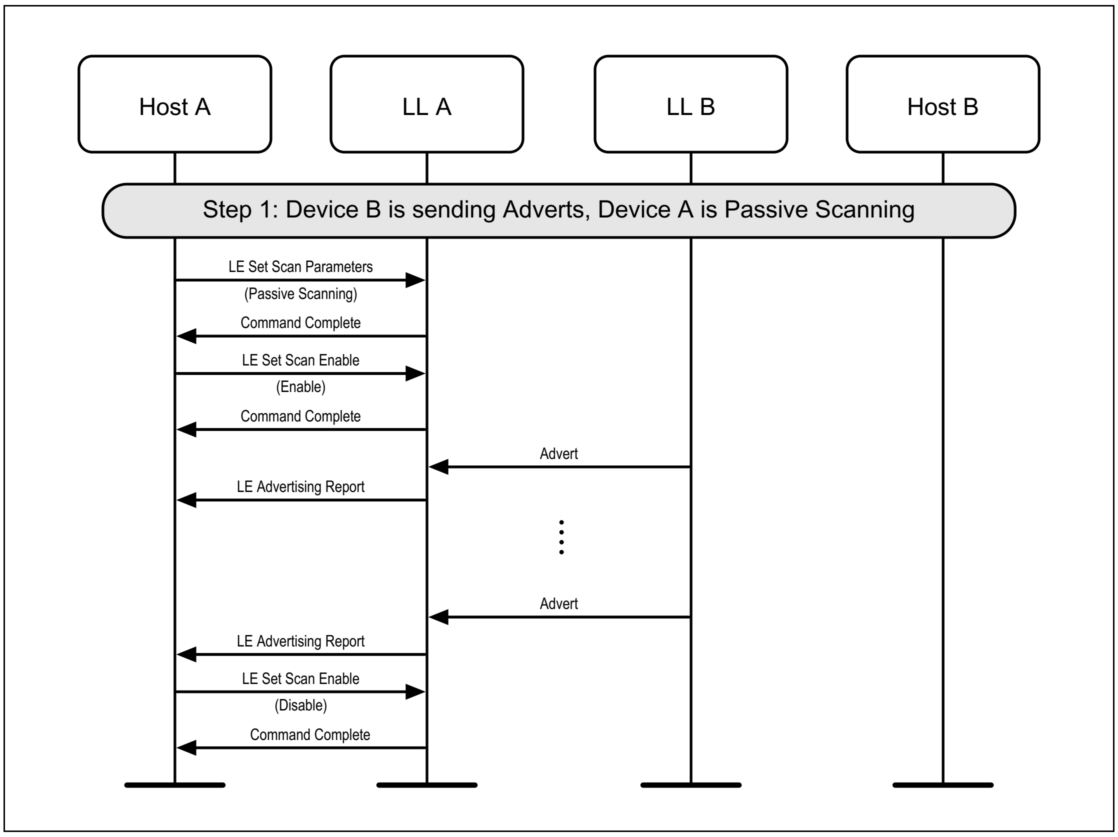


Figure 4.1: Passive Scanning

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).

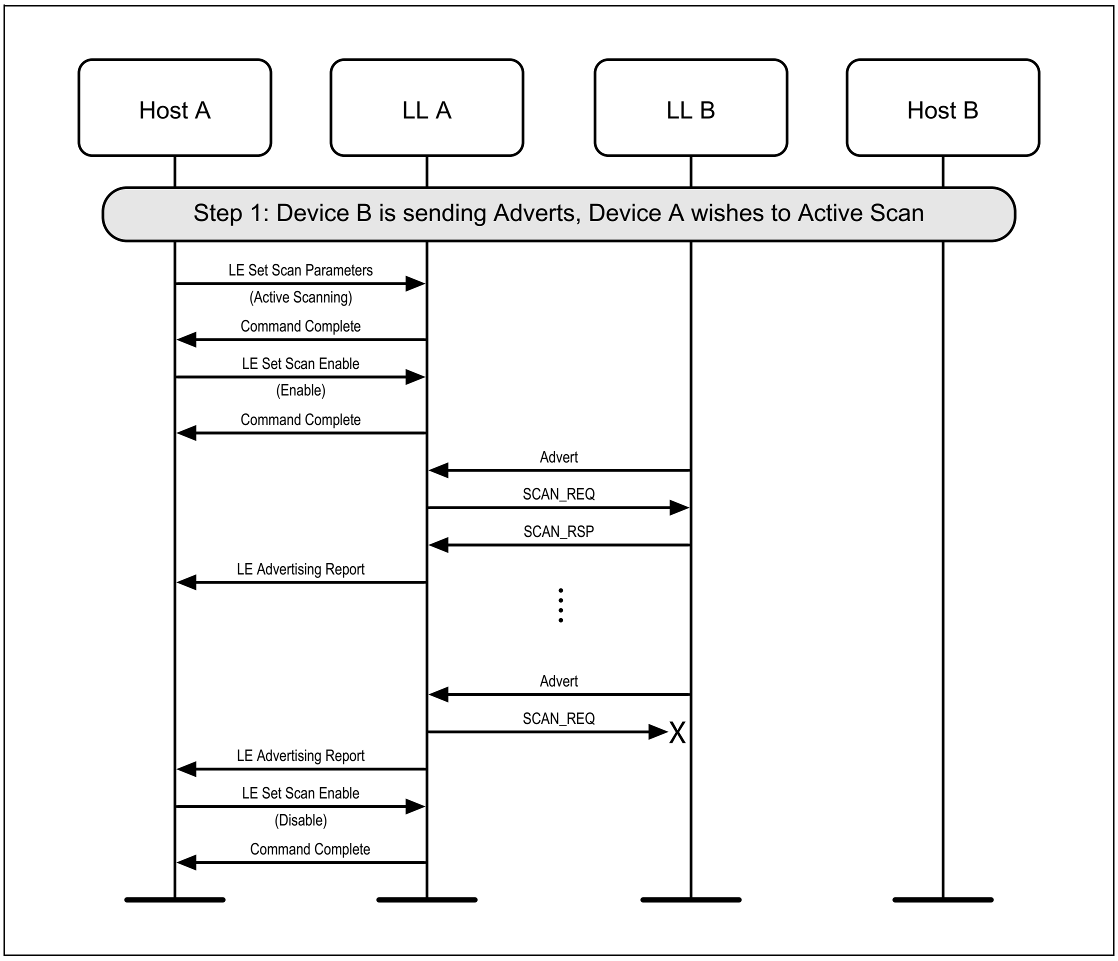


Figure 4.2: Active Scanning

4.3 PASSIVE SCANNING FOR DIRECTED ADVERTISEMENTS WITH PRIVACY

If a device does not support Privacy in the Controller, it may choose to forward LE Directed Advertising Report events from devices supporting Privacy without requiring filtering through the Controller Resolving List.

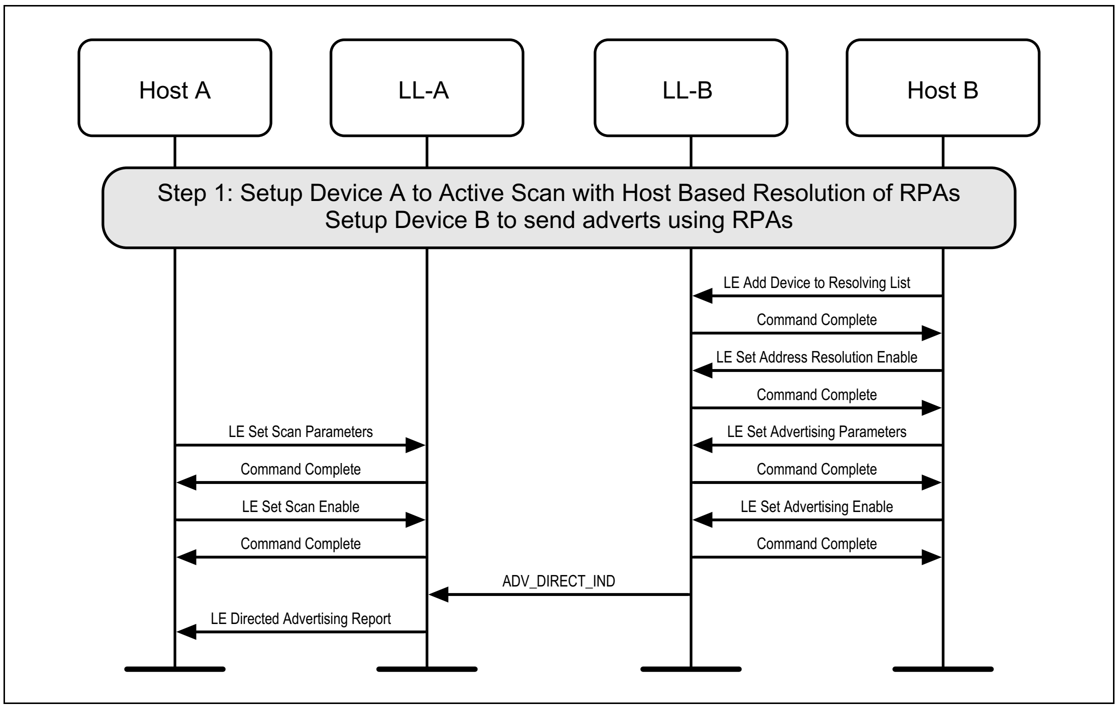

Host A
LL-A
LL-B
Host B
Step 1: Setup Device A to Active Scan with Host Based Resolution of RPAs
Setup Device B to send adverts using RPAs
LE Add Device to Resolving List
Command Complete
LE Set Address Resolution Enable
Command Complete
LE Set Scan Parameters
LE Set Advertising Parameters
Command Complete
Command Complete
LE Set Scan Enable
LE Set Advertising Enable
Command Complete
Command Complete
ADV_DIRECT_IND
LE Directed Advertising Report

Figure 4.3: Directed Advertising with Privacy

4.4 ACTIVE SCANNING WITH PRIVACY

A device may use active scanning to obtain more information about devices that may be useful

to populate a user interface. Privacy may be used during active scanning to make it more

difficult to track either device during active scanning (see Figure 4.4).

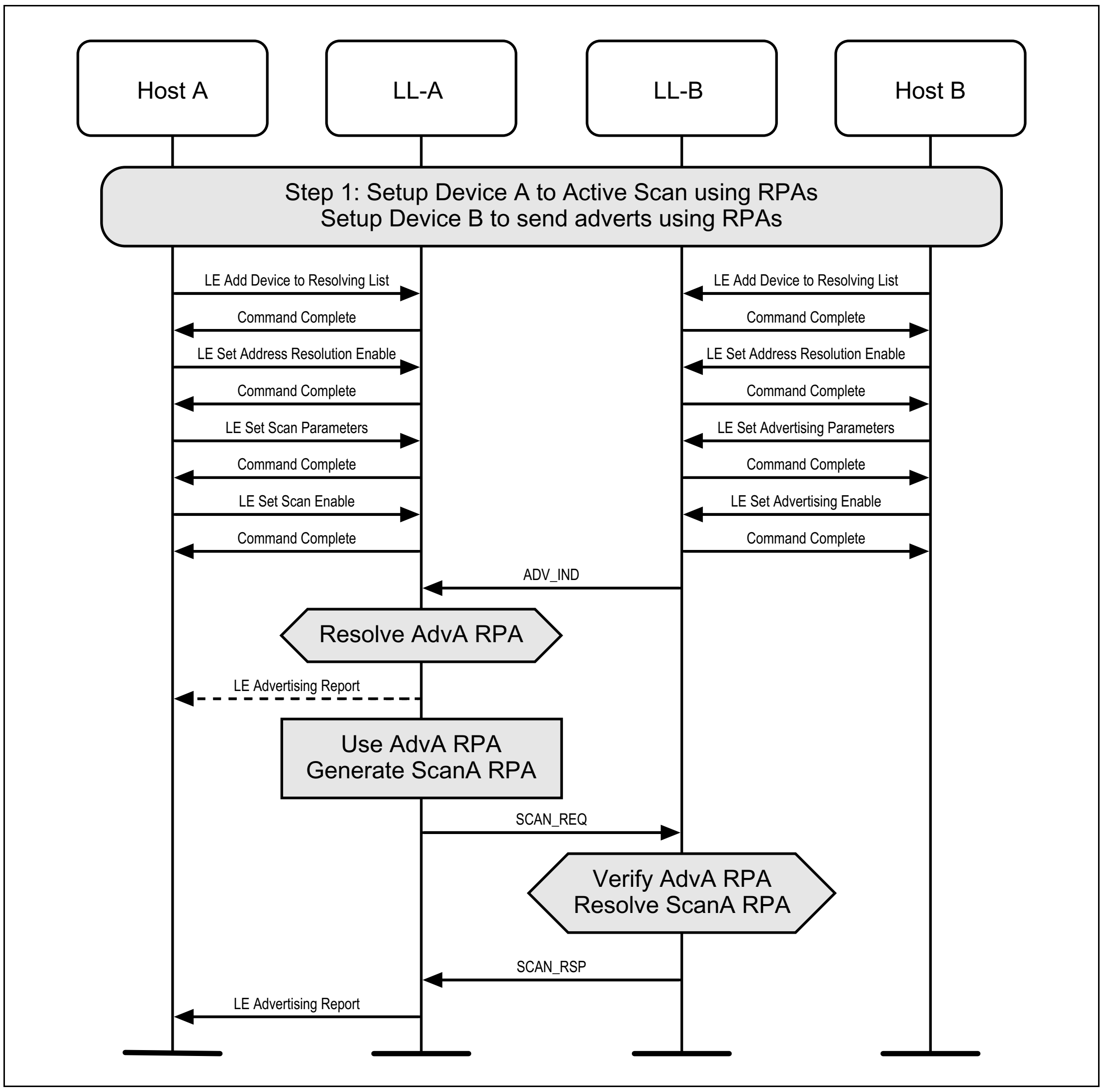


Figure 4.4: Active Scanning with Privacy

4.5 ACTIVE SCANNING WITH PRIVACY AND CONTROLLER BASED RESOLVABLE PRIVATE ADDRESS GENERATION

A Controller will periodically update the resolvable private addresses used on both devices if the devices use active scanning and advertising with Privacy. A Host may at anytime retrieve the read from the Controller the current addresses being used (see Figure 4.5).

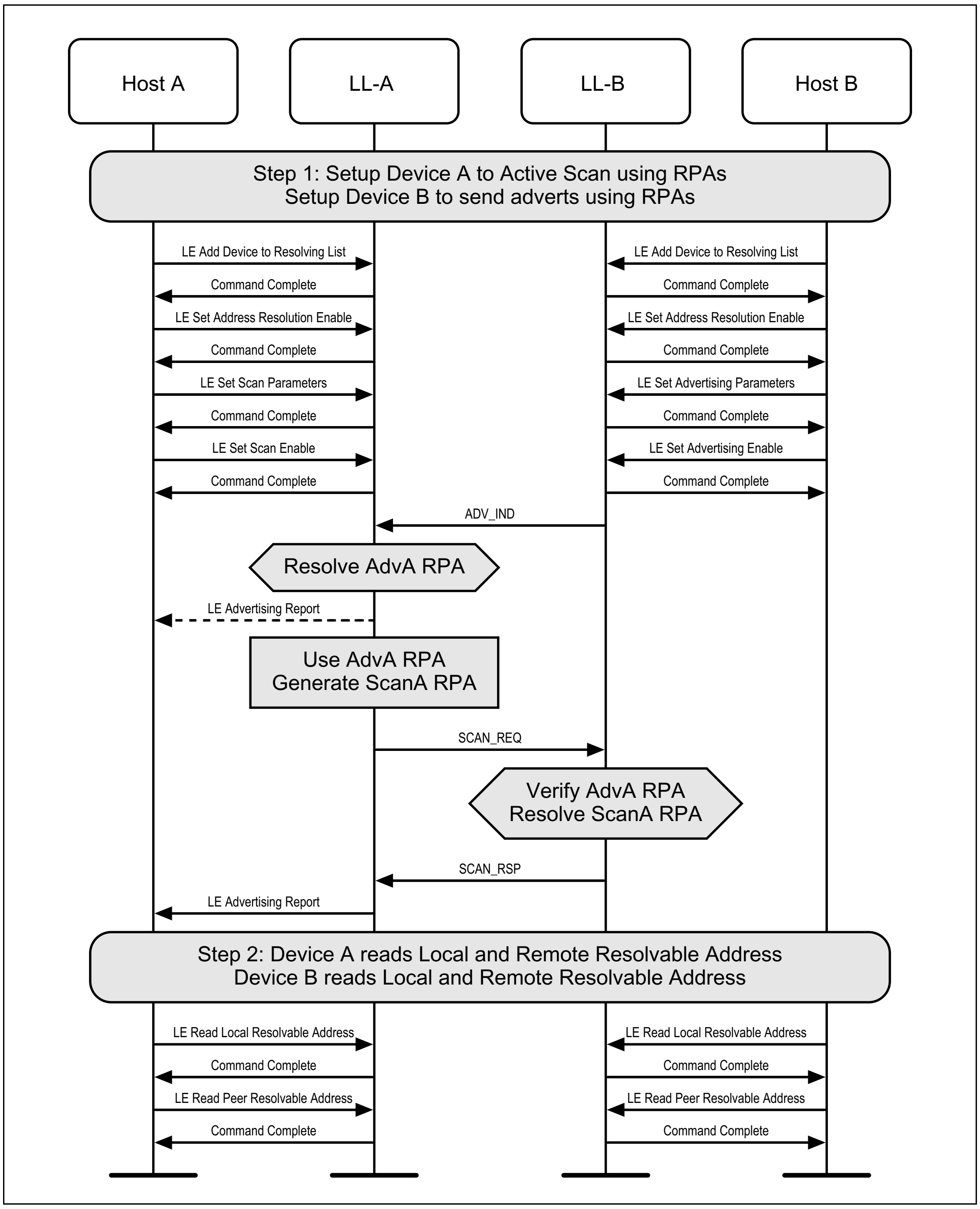

Host A
LL-A
LL-B
Host B
Step 1: Setup Device A to Active Scan using RPAs
Setup Device B to send adverts using RPAs
LE Add Device to Resolving List
Command Complete
LE Set Address Resolution Enable
Command Complete
LE Set Scan Parameters
Command Complete
LE Set Scan Enable
Command Complete
LE Add Device to Resolving List
Command Complete
LE Set Address Resolution Enable
Command Complete
LE Set Advertising Parameters
Command Complete
LE Set Advertising Enable
Command Complete
ADV_IND
Resolve AdvA RPA
LE Advertising Report
Use AdvA RPA
Generate ScanA RPA
SCAN_REQ
Verify AdvA RPA
Resolve ScanA RPA
SCAN_RSP
LE Advertising Report
Step 2: Device A reads Local and Remote Resolvable Address
Device B reads Local and Remote Resolvable Address
LE Read Local Resolvable Address
Command Complete
LE Read Peer Resolvable Address
Command Complete
LE Read Local Resolvable Address
Command Complete
LE Read Peer Resolvable Address
Command Complete

Figure 4.5: Retrieving local and remote resolvable address updates from the Controller

4.6 ACTIVE SCANNING ON THE SECONDARY ADVERTISING CHANNEL

A device may use active scanning on the secondary advertising channel in order to obtain more information about devices that may be useful to populate a user interface (see Figure 4.6).

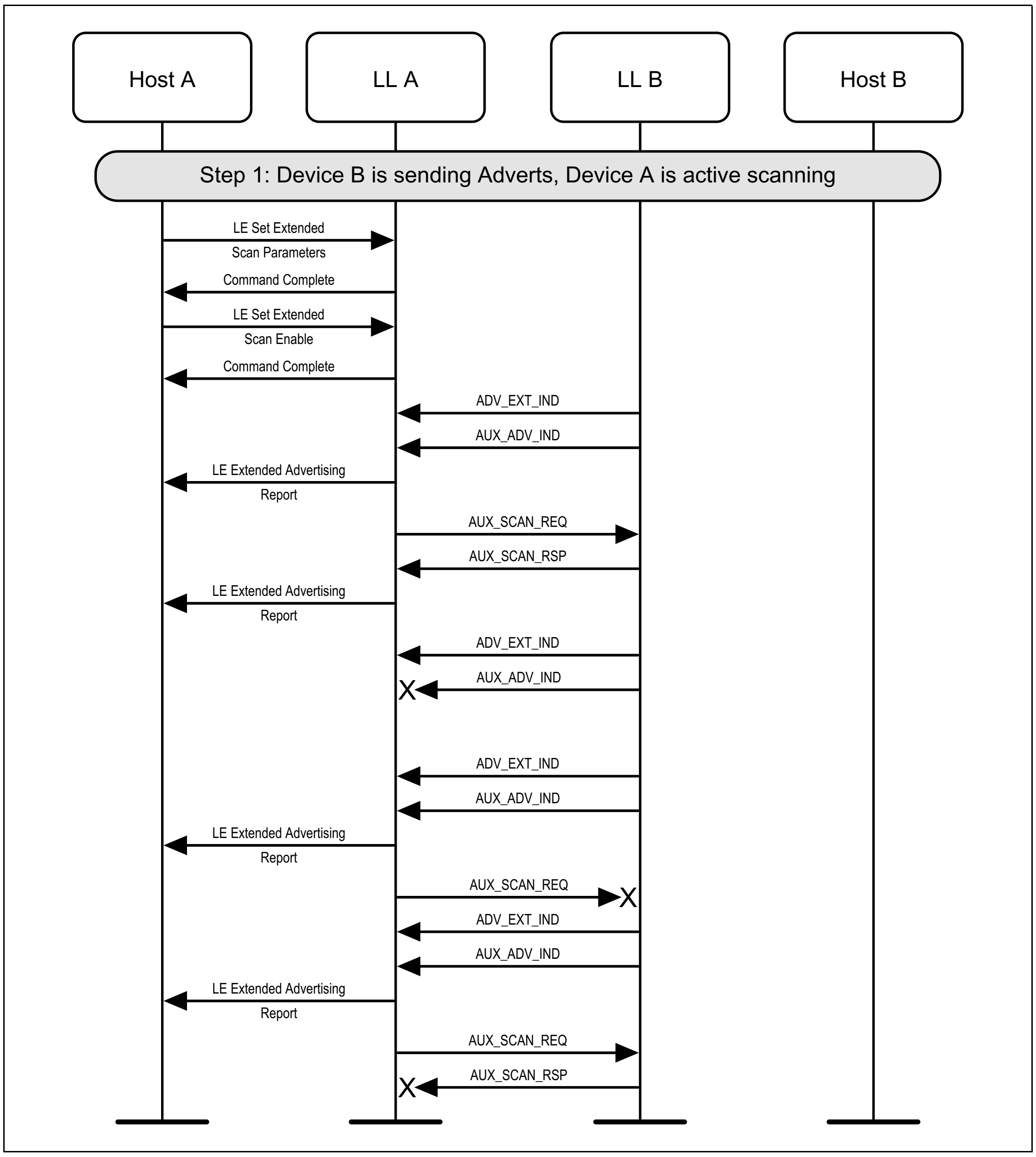

Host A
LL A
LL B
Host B
Step 1: Device B is sending Adverts, Device A is active scanning
LE Set Extended Scan Parameters
Command Complete
LE Set Extended Scan Enable
Command Complete
ADV_EXT_IND
AUX_ADV_IND
LE Extended Advertising Report
AUX_SCAN_REQ
AUX_SCAN_RSP
LE Extended Advertising Report
ADV_EXT_IND
AUX_ADV_IND
ADV_EXT_IND
AUX_ADV_IND
LE Extended Advertising Report
AUX_SCAN_REQ
ADV_EXT_IND
AUX_ADV_IND
LE Extended Advertising Report
AUX_SCAN_REQ
AUX_SCAN_RSP

Figure 4.6: Extended active scanning on the secondary advertising channel

4.7 SCAN TIMEOUT

A device may scan for a limited duration of time (see Figure 4.7).

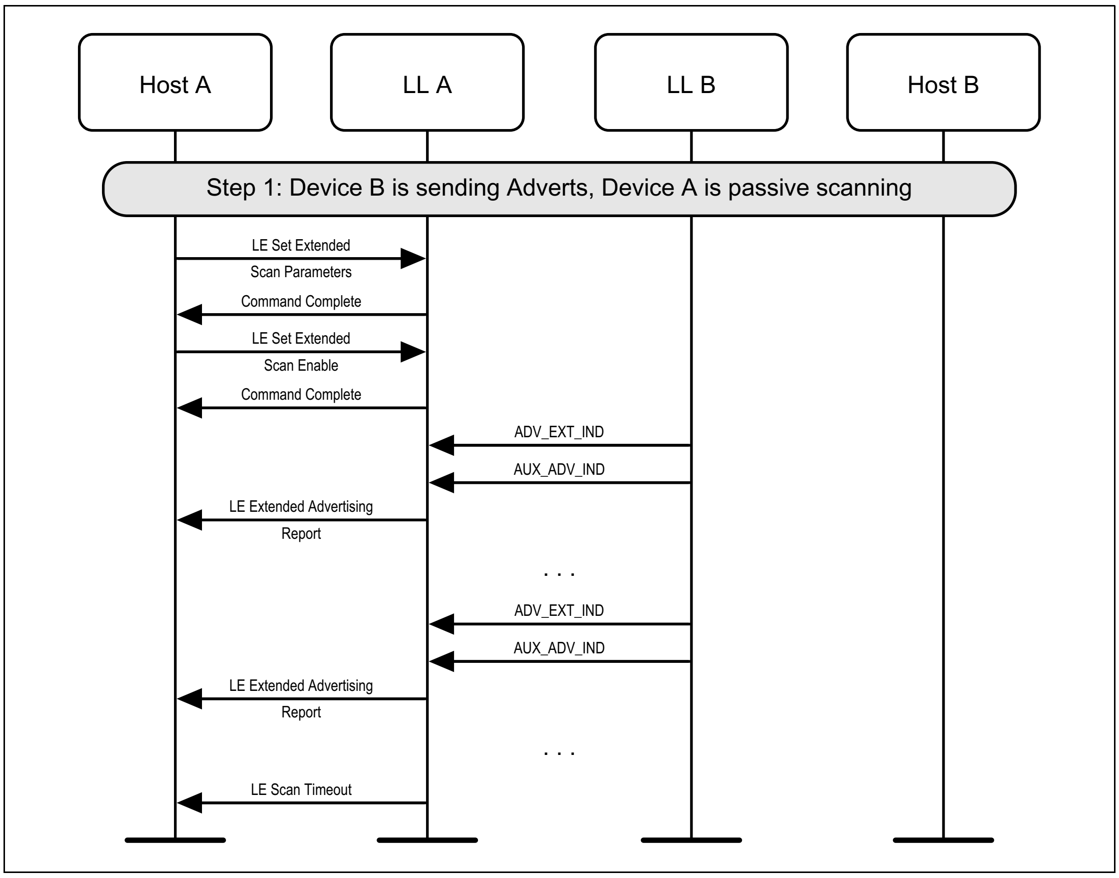
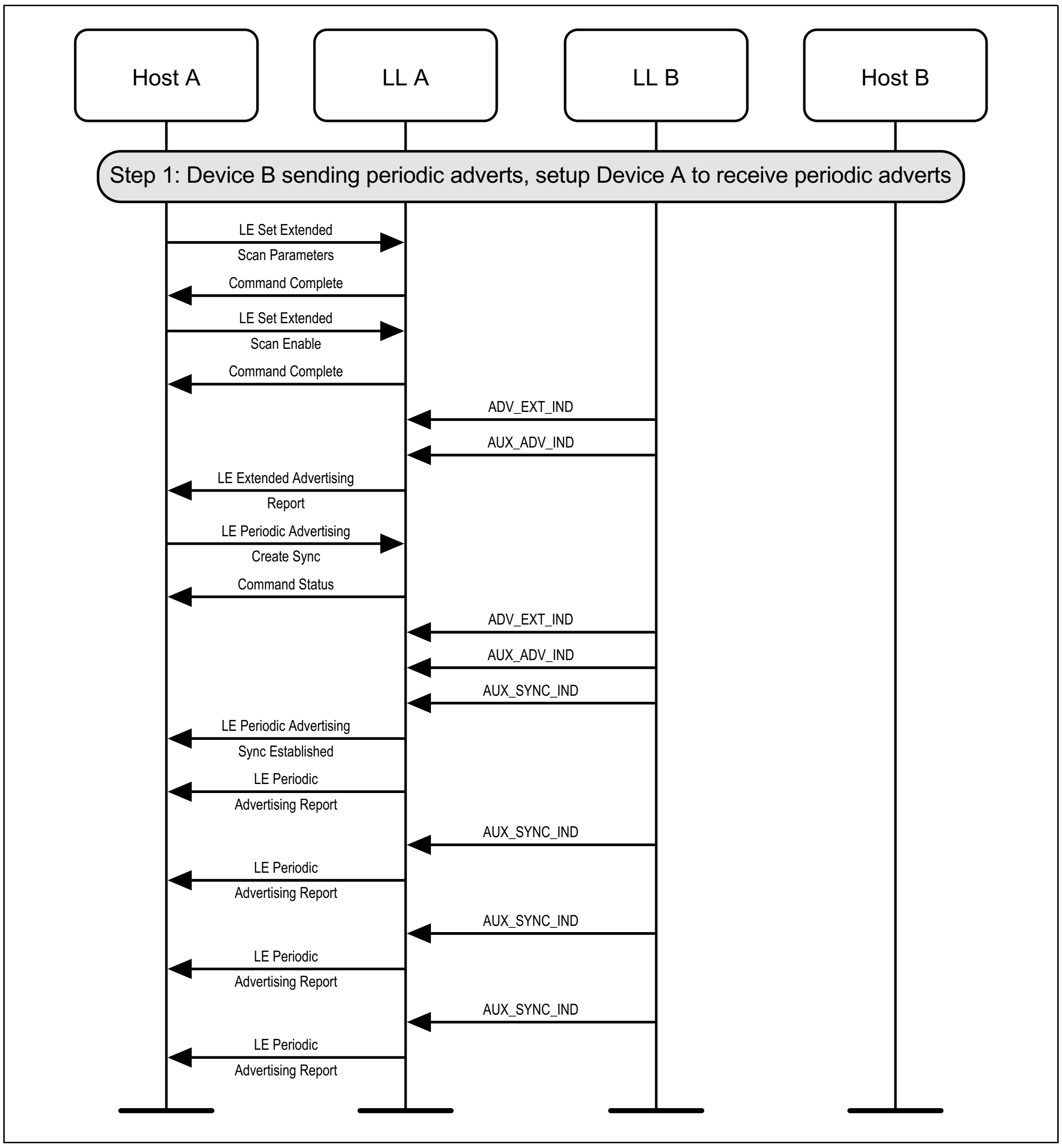

Host A
LL A
LL B
Host B
Step 1: Device B is sending Adverts, Device A is passive scanning
LE Set Extended Scan Parameters
Command Complete
LE Set Extended Scan Enable
Command Complete
ADV_EXT_IND
AUX_ADV_IND
LE Extended Advertising Report
. . .
ADV_EXT_IND
AUX_ADV_IND
LE Extended Advertising Report
. . .
LE Scan Timeout

Figure 4.7: Scan Timeout

4.8 PERIODIC SCANNING

A device may establish synchronization with a periodic advertiser and report periodic advertising packets to the Host (see Figure 4.8).



Host A
LL A
LL B
Host B
Step 1: Device B sending periodic adverts, setup Device A to receive periodic adverts
LE Set Extended Scan Parameters
Command Complete
LE Set Extended Scan Enable
Command Complete
ADV_EXT_IND
AUX_ADV_IND
LE Extended Advertising Report
LE Periodic Advertising Create Sync
Command Status
ADV_EXT_IND
AUX_ADV_IND
AUX_SYNC_IND
LE Periodic Advertising Sync Established
LE Periodic Advertising Report
AUX_SYNC_IND
LE Periodic Advertising Report
AUX_SYNC_IND
LE Periodic Advertising Report
AUX_SYNC_IND
LE Periodic Advertising Report

Figure 4.8: Periodic Scanning

4.9 PERIODIC SCANNING CANCEL

A device may cancel a pending request to establish synchronization with a periodic advertiser. This example shows an unsuccessful synchronization, followed by cancellation of the synchronization (see Figure 4.9).

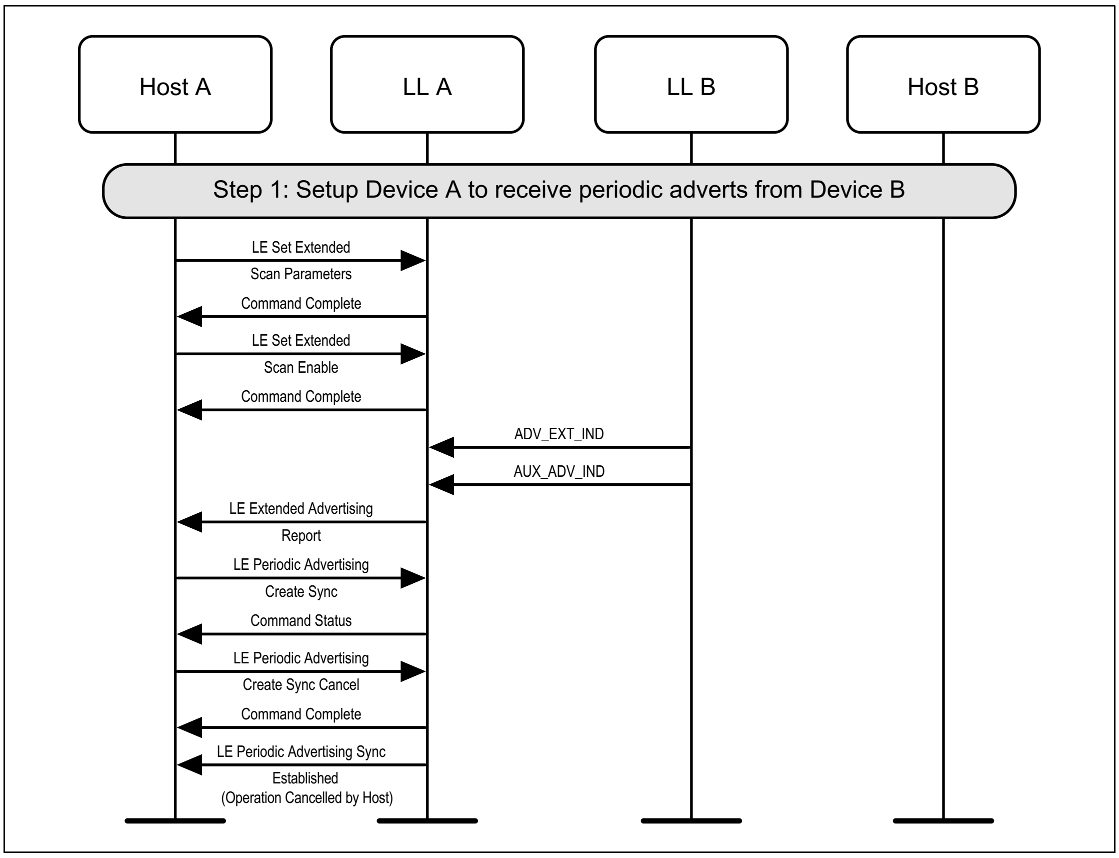

Host A
LL A
LL B
Host B
Step 1: Setup Device A to receive periodic adverts from Device B
LE Set Extended Scan Parameters
Command Complete
LE Set Extended Scan Enable
Command Complete
ADV_EXT_IND
AUX_ADV_IND
LE Extended Advertising Report
LE Periodic Advertising Create Sync
Command Status
LE Periodic Advertising Create Sync Cancel
Command Complete
LE Periodic Advertising Sync Established (Operation Cancelled by Host)

Figure 4.9: Periodic Scanning Cancel

4.10 PERIODIC SCANNING TIMEOUT

A device may lose synchronization with a periodic advertiser (see Figure 4.10).

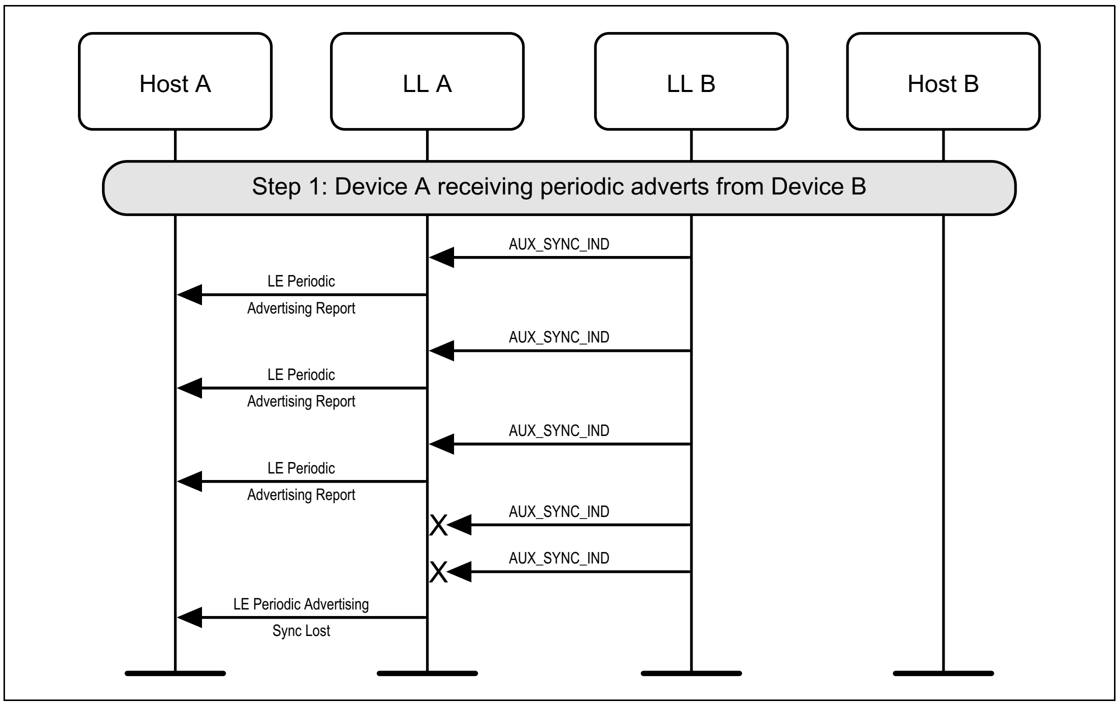

Host A
LL A
LL B
Host B
Step 1: Device A receiving periodic adverts from Device B
AUX_SYNC_IND
LE Periodic Advertising Report
AUX_SYNC_IND
LE Periodic Advertising Report
AUX_SYNC_IND
LE Periodic Advertising Report
AUX_SYNC_IND
AUX_SYNC_IND
LE Periodic Advertising Sync Lost

Figure 4.10: Periodic Scanning Timeout

4.11 PERIODIC SCANNING TERMINATE

Once synchronized with a periodic advertiser, the Host can terminate the synchronization (see Figure 4.11).

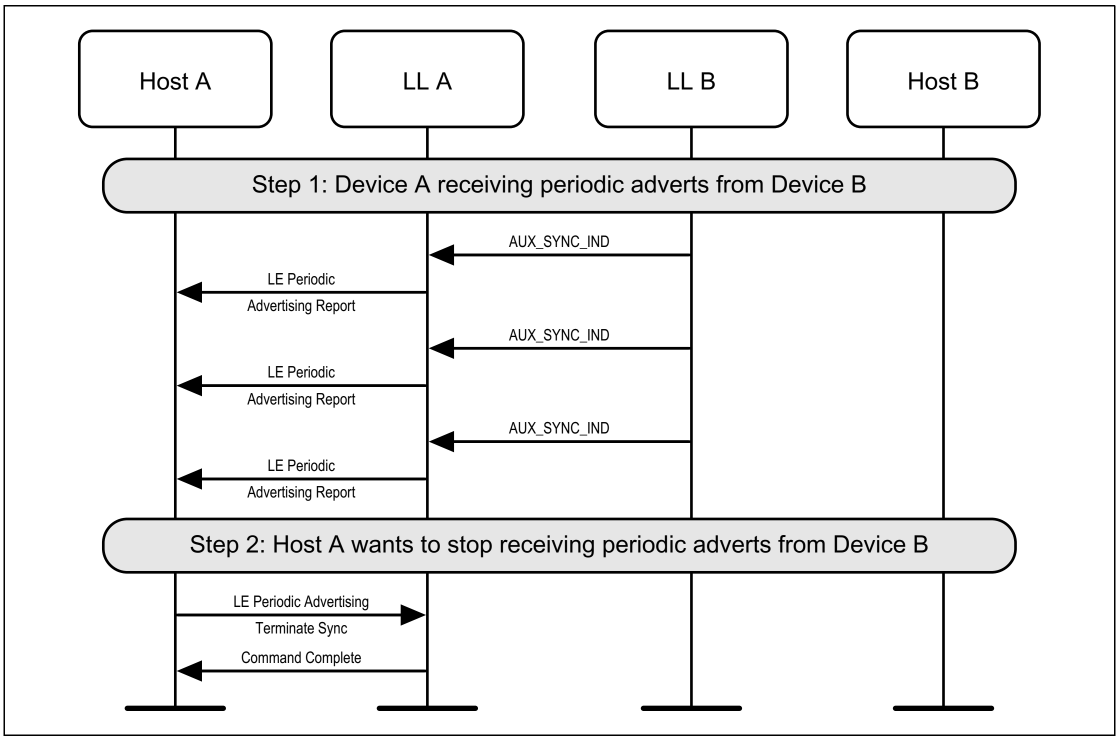

Host A
LL A
LL B
Host B
Step 1: Device A receiving periodic adverts from Device B
AUX_SYNC_IND
LE Periodic Advertising Report
AUX_SYNC_IND
LE Periodic Advertising Report
AUX_SYNC_IND
LE Periodic Advertising Report
Step 2: Host A wants to stop receiving periodic adverts from Device B
LE Periodic Advertising Terminate Sync
Command Complete

Figure 4.11: Periodic Scanning Terminate

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).

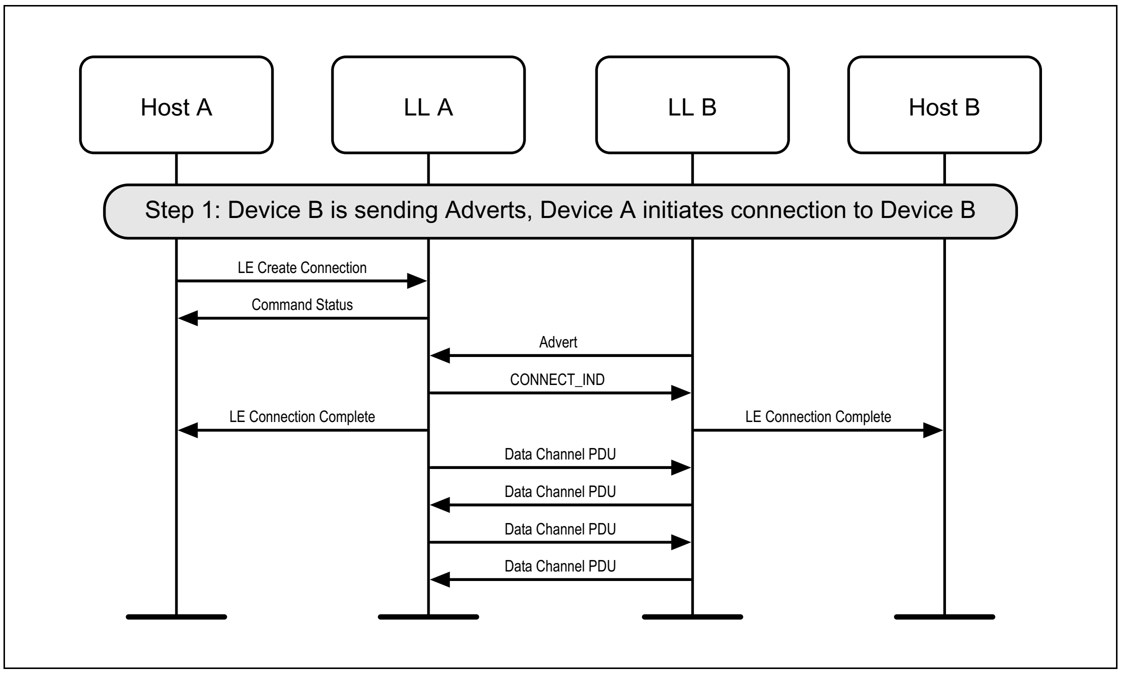


Figure 5.1: Initiating a Connection

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).

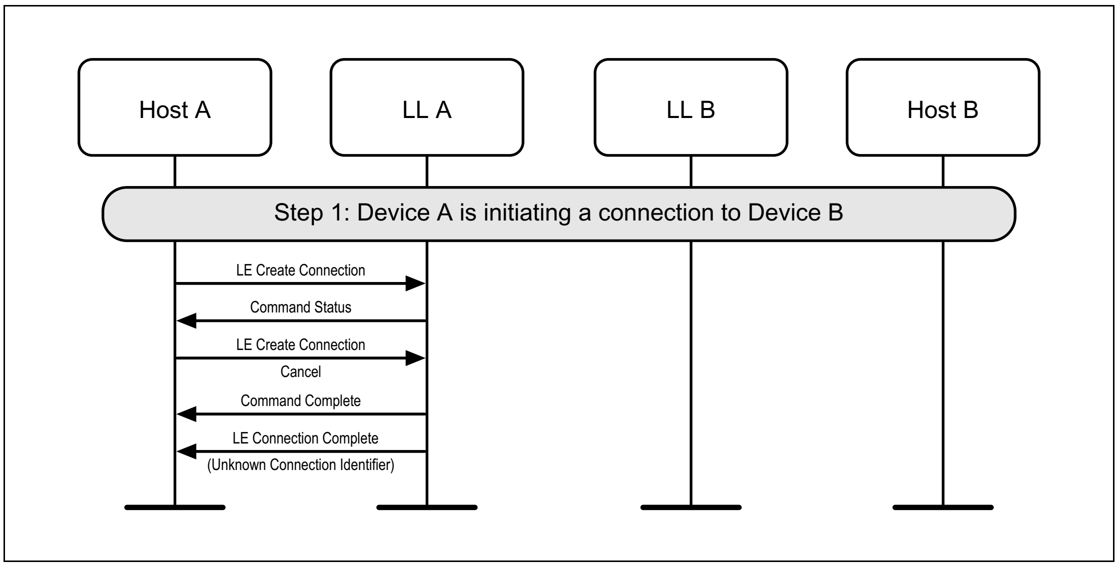

Host A
LL A
LL B
Host B
Step 1: Device A is initiating a connection to Device B
LE Create Connection
Command Status
LE Create Connection Cancel
Command Complete
LE Connection Complete (Unknown Connection Identifier)

Figure 5.2: Canceling an Initiation

5.3 INITIATING A CONNECTION USING UNDIRECTED ADVERTISING WITH PRIVACY

A device can initiate a connection to an advertiser. Privacy may be used during connection initiation to make it more difficult to track either device during connection setup. The example shows a successful initiation, resulting in both devices able to send application data (see Figure 5.3).

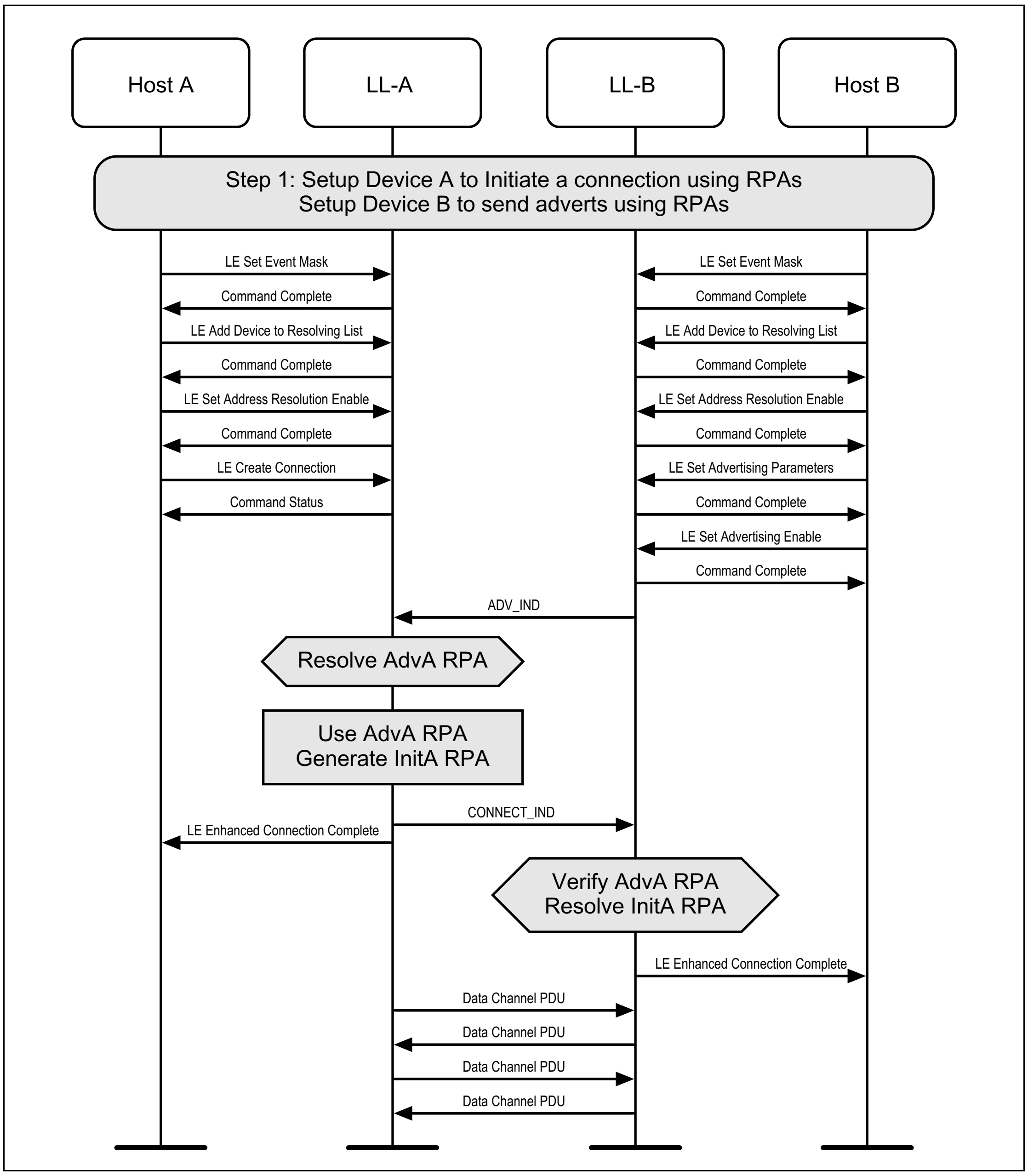


Figure 5.3: Initiating a connection using Undirected Advertising with Privacy

5.4 INITIATING A CONNECTION USING DIRECTED ADVERTISING WITH PRIVACY

A device can initiate a connection to an advertiser who is using Directed Advertising. Privacy may be used during connection initiation to make it more difficult to track either device during connection setup as well as target a single initiator. The example shows a successful initiation, resulting in both devices able to send application data (see Figure 5.4).

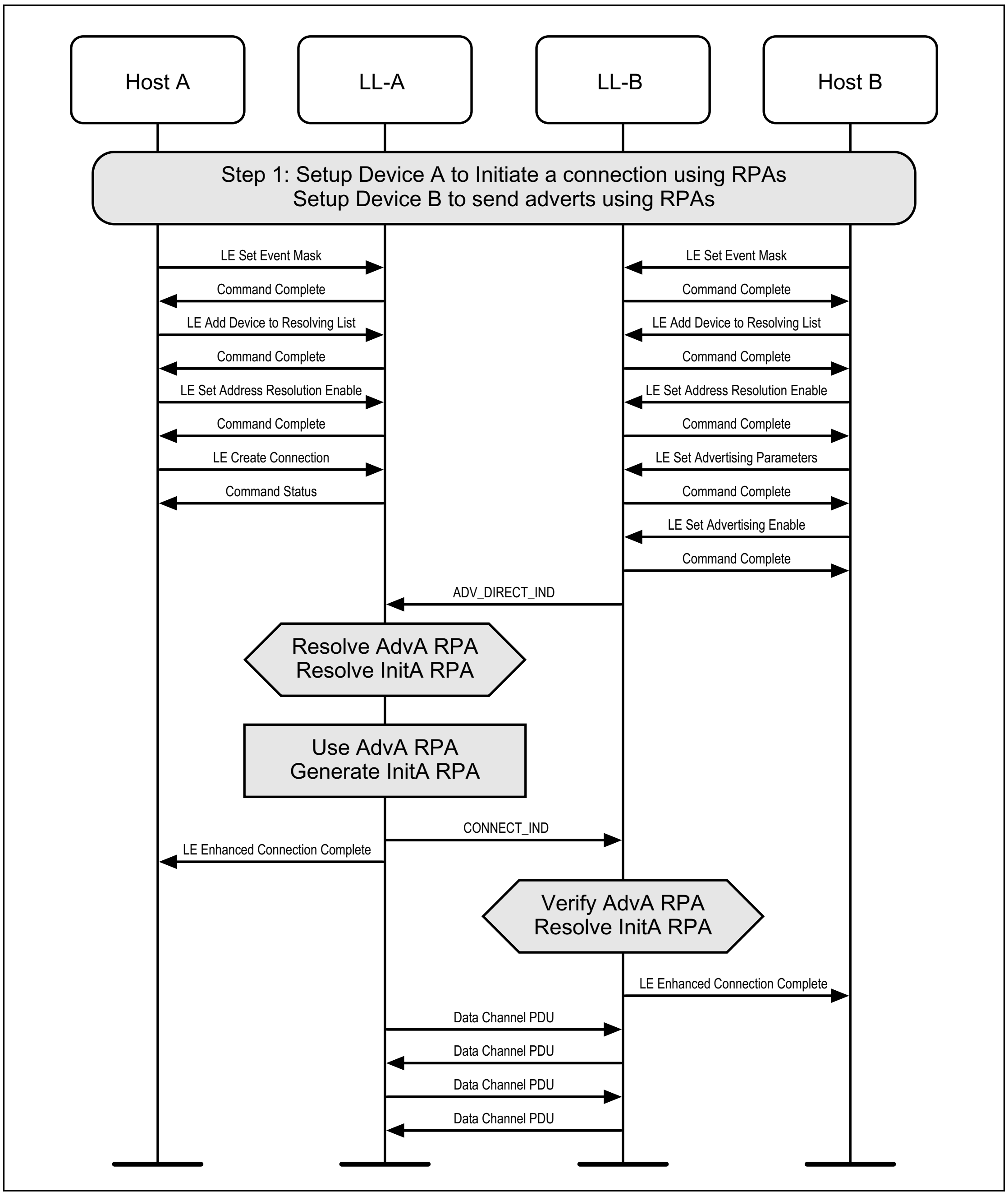


Figure 5.4: Initiating a connection using Directed Advertising with Privacy

5.5 INITIATING A CONNECTION THAT FAILS TO ESTABLISH

This example shows an initiation that fails to establish because Device B (the advertiser) fails to respond to the Data Channel PDUs sent by Device A.

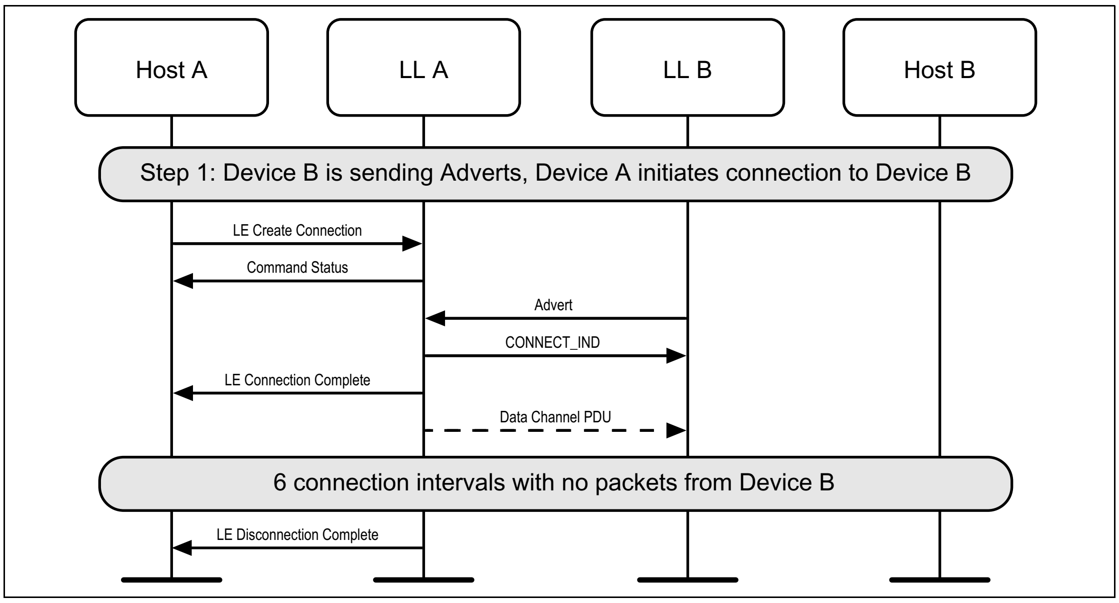

Host A
LL A
LL B
Host B
Step 1: Device B is sending Adverts, Device A initiates connection to Device B
LE Create Connection
Command Status
Advert
CONNECT_IND
LE Connection Complete
Data Channel PDU
6 connection intervals with no packets from Device B
LE Disconnection Complete

Figure 5.5: Initiating a Connection that fails to establish

Device A may or may not send data channel PDUs in the 6 connection intervals before establishment fails. If it does not do so, Device B is unable to respond.

5.6 INITIATING A CONNECTION ON THE SECONDARY ADVERTISING CHANNEL

A device can initiate a connection to an advertiser on the secondary channel. This example shows a successful initiation, resulting in both devices able to send application data (see Figure 5.6).

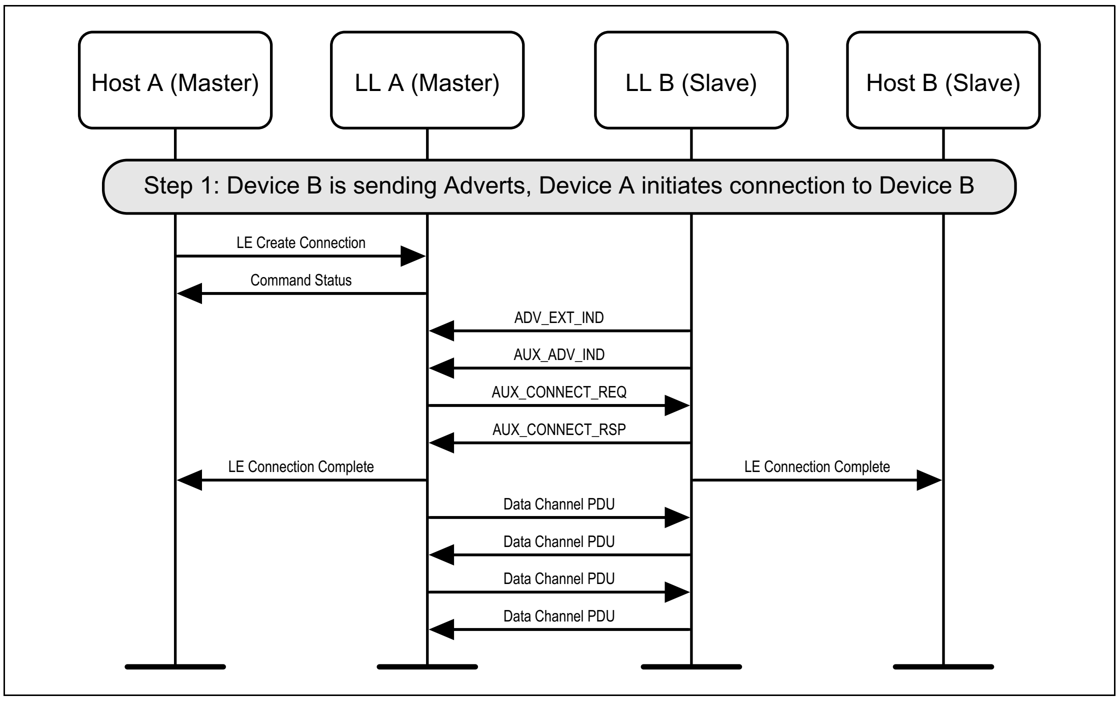


Figure 5.6: Initiating a connection on the secondary advertising channel

5.7 INITIATING A CHANNEL SELECTION ALGORITHM #2 CONNECTION

Where a device supports the Channel Selection Algorithm #2 feature, it can initiate a connection which will use Channel Selection Algorithm #2 to an advertiser who has the ChSel field of the advertising channel PDU set to 1. The example shows a successful initiation, resulting in the connection using Channel Selection Algorithm #2.

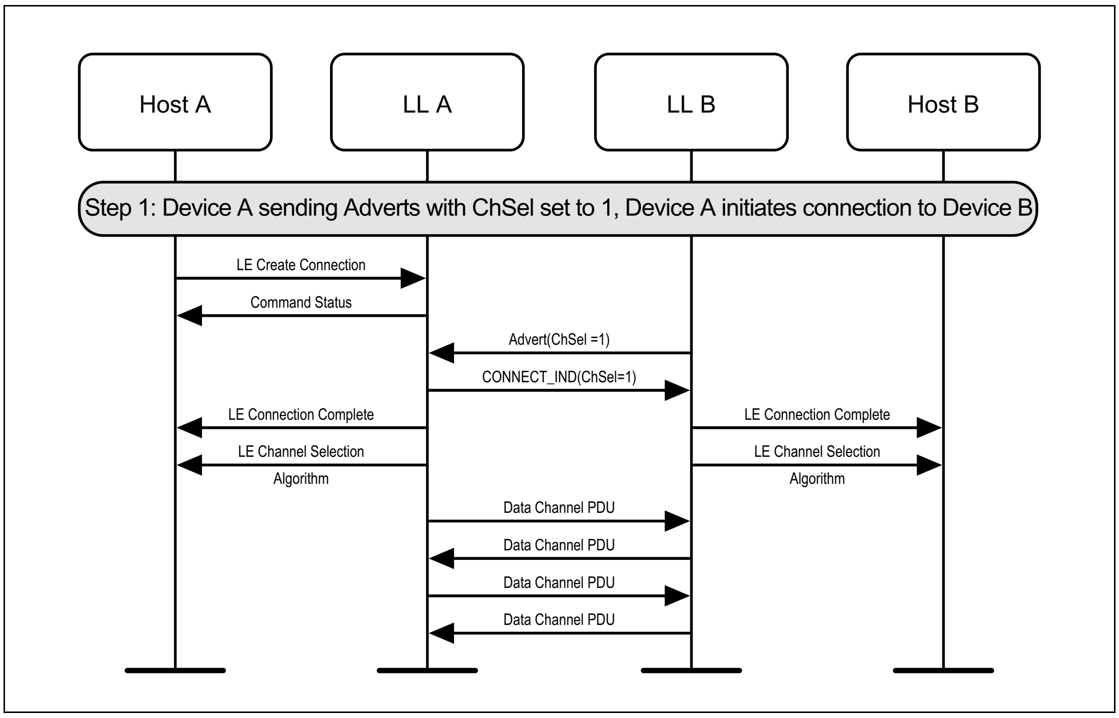


Figure 5.7: Initiating a Channel Selection Algorithm #2 Connection

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).

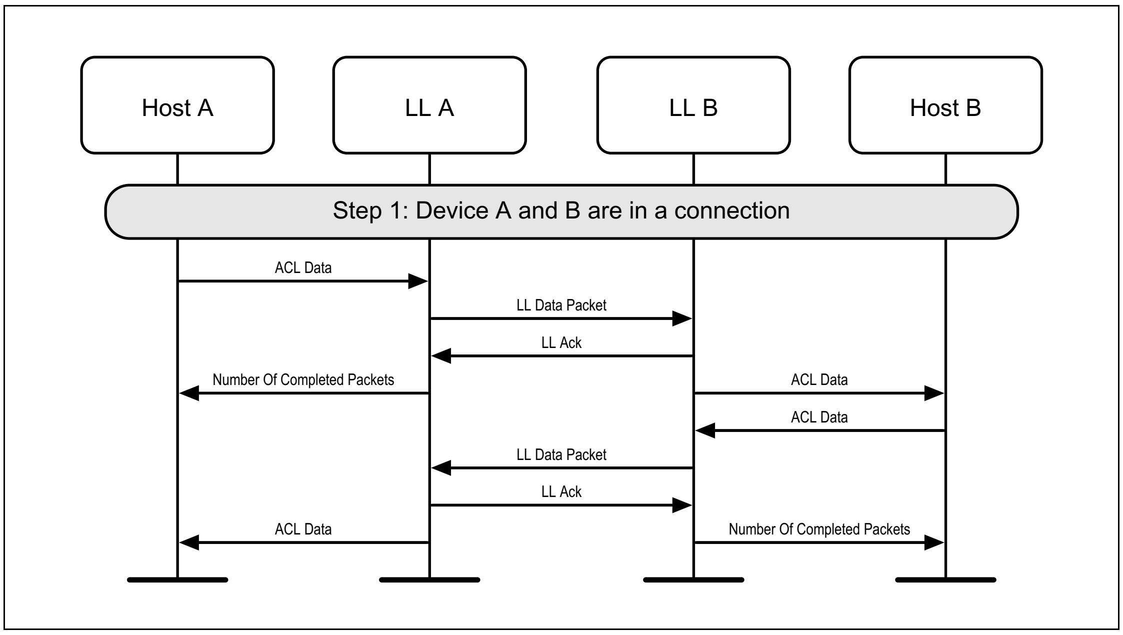

Host A
LL A
LL B
Host B
Step 1: Device A and B are in a connection
ACL Data
LL Data Packet
LL Ack
Number Of Completed Packets
ACL Data
ACL Data
LL Data Packet
LL Ack
ACL Data
Number Of Completed Packets

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).

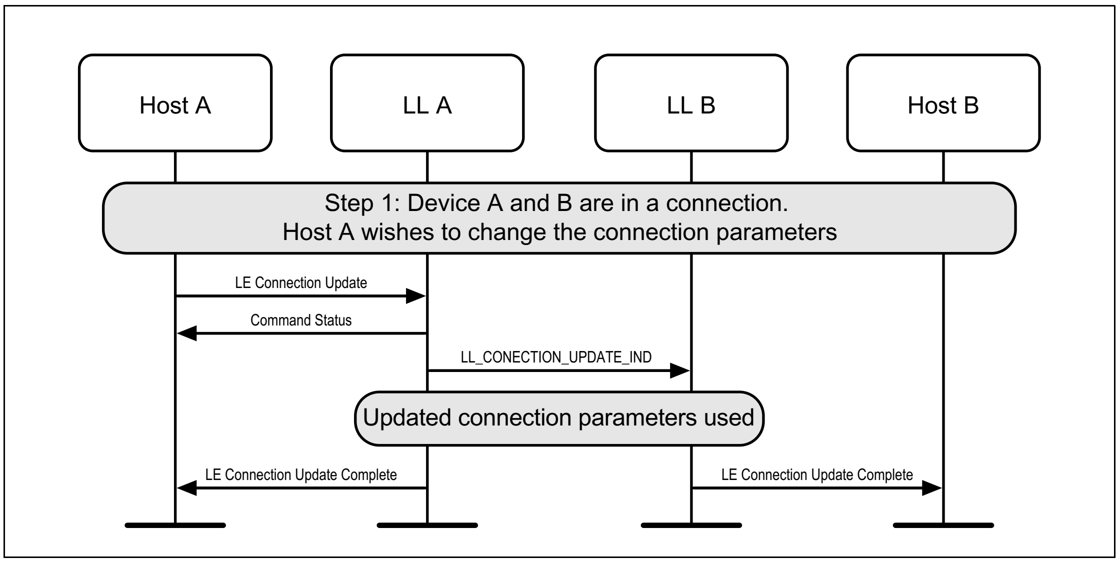

Host A
LL A
LL B
Host B
Step 1: Device A and B are in a connection.
Host A wishes to change the connection parameters
LE Connection Update
Command Status
LL_CONECTION_UPDATE_IND
Updated connection parameters used
LE Connection Update Complete
LE Connection Update Complete

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).

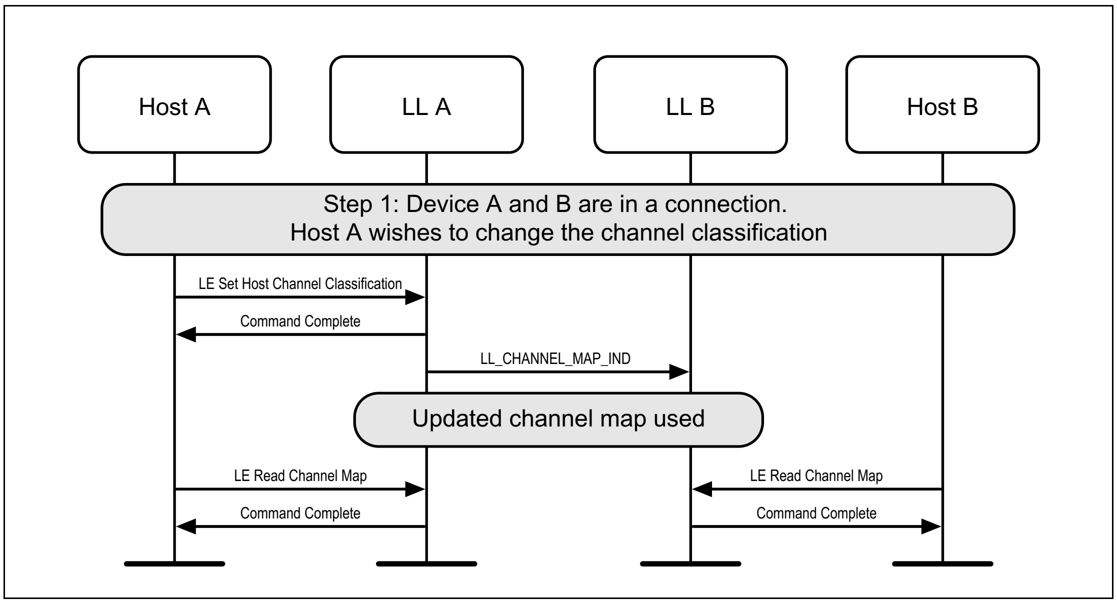


Figure 6.3: Channel Map Update

6.4 FEATURES EXCHANGE

Both the master and slave devices can discover the set of features available on the remote device. To achieve this, the Feature Exchange Link Layer Control Procedure is used (see Figure 6.4 and Figure 6.5).

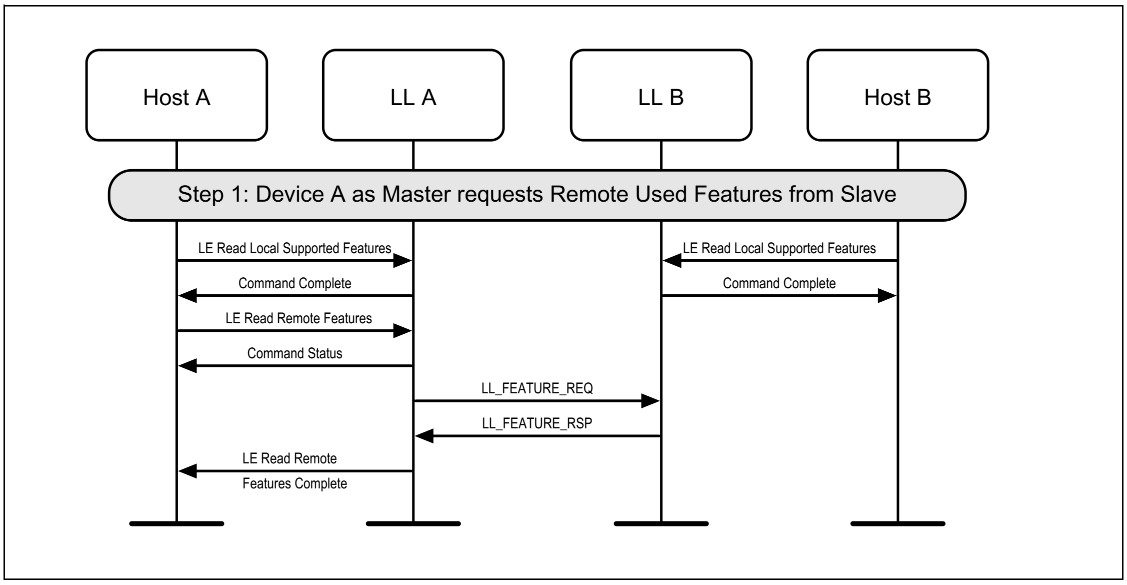

Host A
LL A
LL B
Host B
Step 1: Device A as Master requests Remote Used Features from Slave
LE Read Local Supported Features
LE Read Local Supported Features
Command Complete
Command Complete
LE Read Remote Features
Command Status
LL_FEATURE_REQ
LL_FEATURE_RSP
LE Read Remote Features Complete

Figure 6.4: Master-initiated Features Exchange

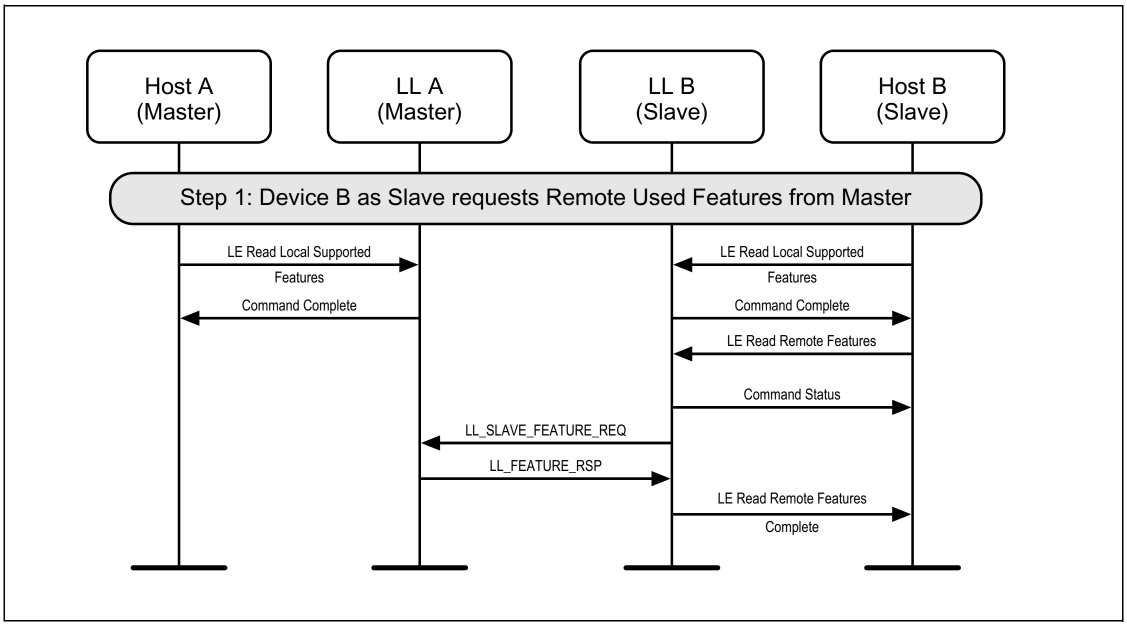

Host A (Master)
LL A (Master)
LL B (Slave)
Host B (Slave)
Step 1: Device B as Slave requests Remote Used Features from Master
LE Read Local Supported Features
LE Read Local Supported Features
Command Complete
Command Complete
LE Read Remote Features
Command Status
LL_SLAVE_FEATURE_REQ
LL_FEATURE_RSP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

Figure 6.5: Slave-initiated Features Exchange

6.5 VERSION EXCHANGE

Either device may perform a version exchange (see Figure 6.6 and Figure 6.7).

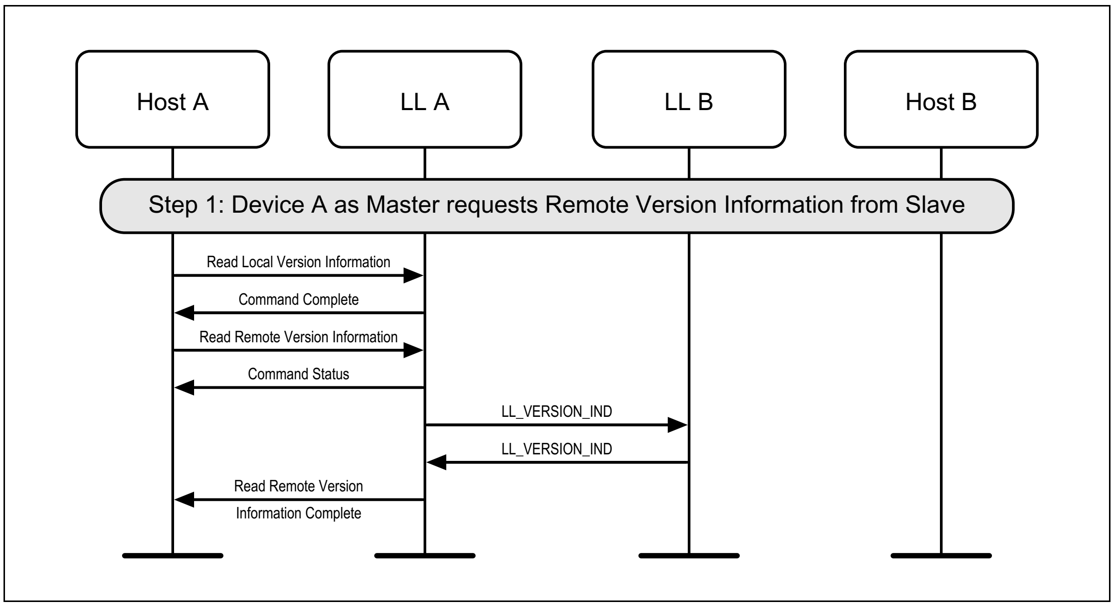

Host A
LL A
LL B
Host B
Step 1: Device A as Master requests Remote Version Information from Slave
Read Local Version Information
Command Complete
Read Remote Version Information
Command Status
LL_VERSION_IND
LL_VERSION_IND
Read Remote Version Information Complete

Figure 6.6: Version Exchange from Master

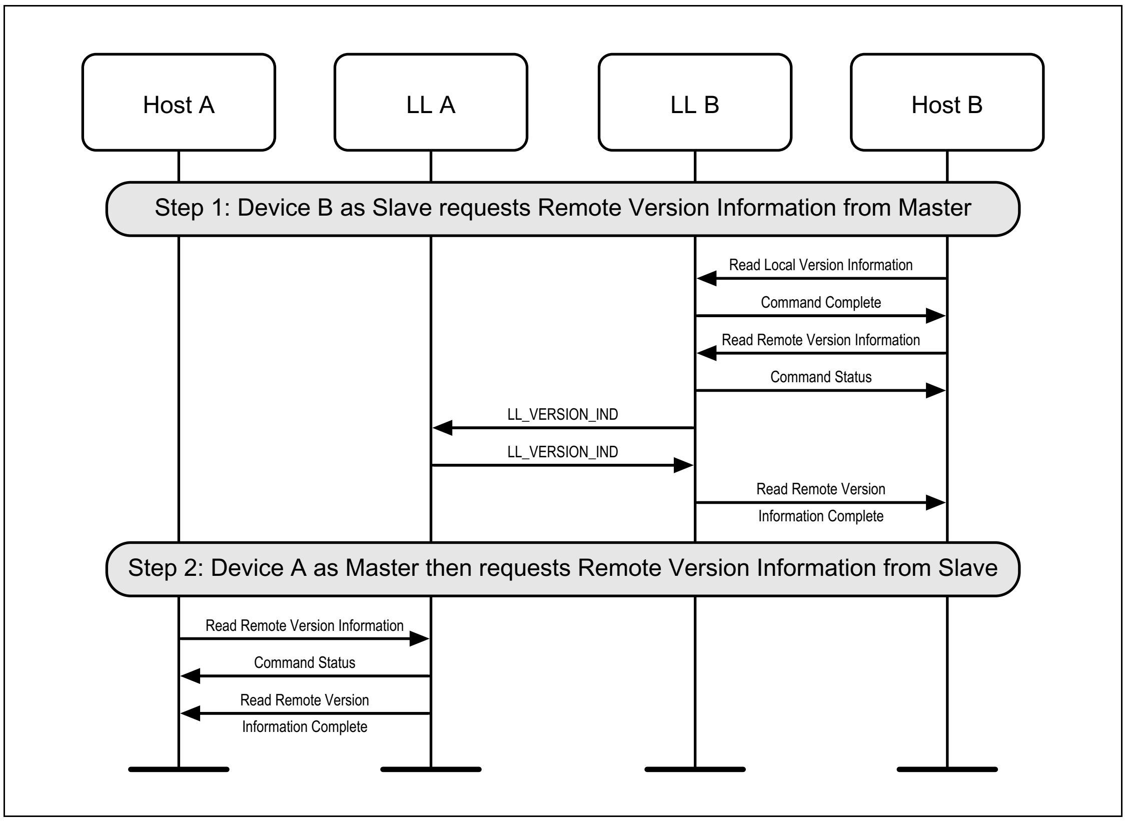

Host A
LL A
LL B
Host B
Step 1: Device B as Slave requests Remote Version Information from Master
Read Local Version Information
Command Complete
Read Remote Version Information
Command Status
LL_VERSION_IND
LL_VERSION_IND
Read Remote Version Information Complete
Step 2: Device A as Master then requests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

Figure 6.7: 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 6.8).

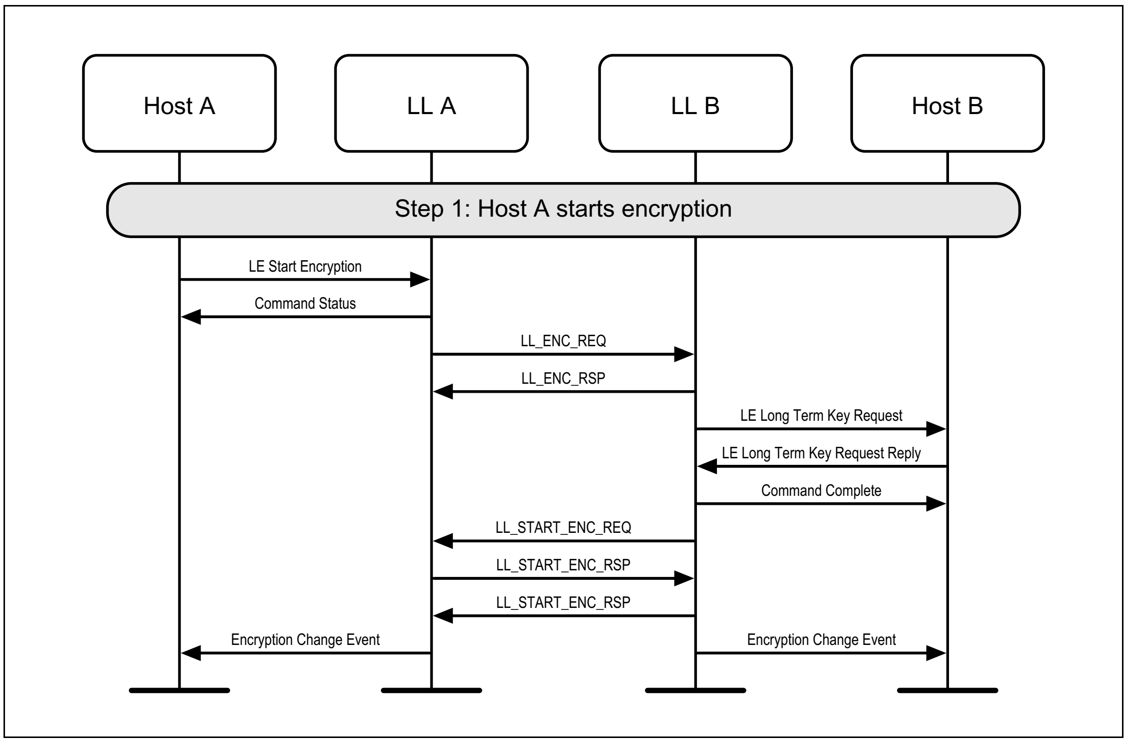


Figure 6.8: Start Encryption

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 6.9 shows the failure case of the slave not having the long term key for the master.

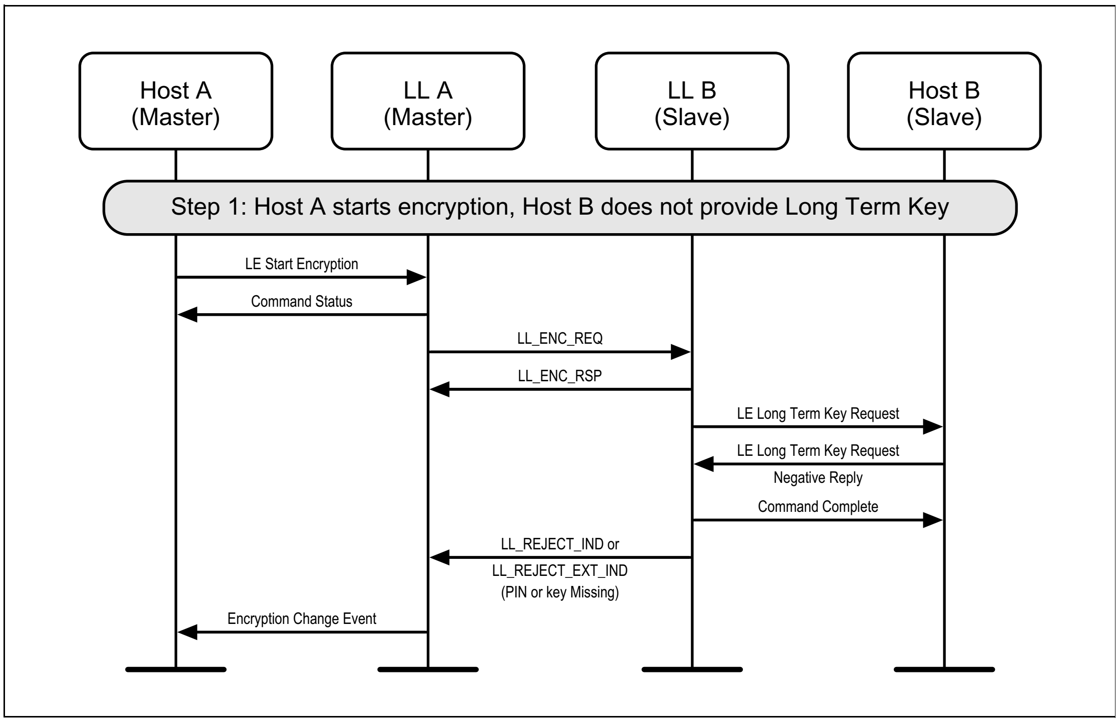


Figure 6.9: Start encryption without long-term key

6.8 START ENCRYPTION WITH EVENT MASKED

If encryption has not been started on a connection, it may be started by the master. Figure 6.10 shows the failure case when the slave has masked out the LE Long Term Key Request event.

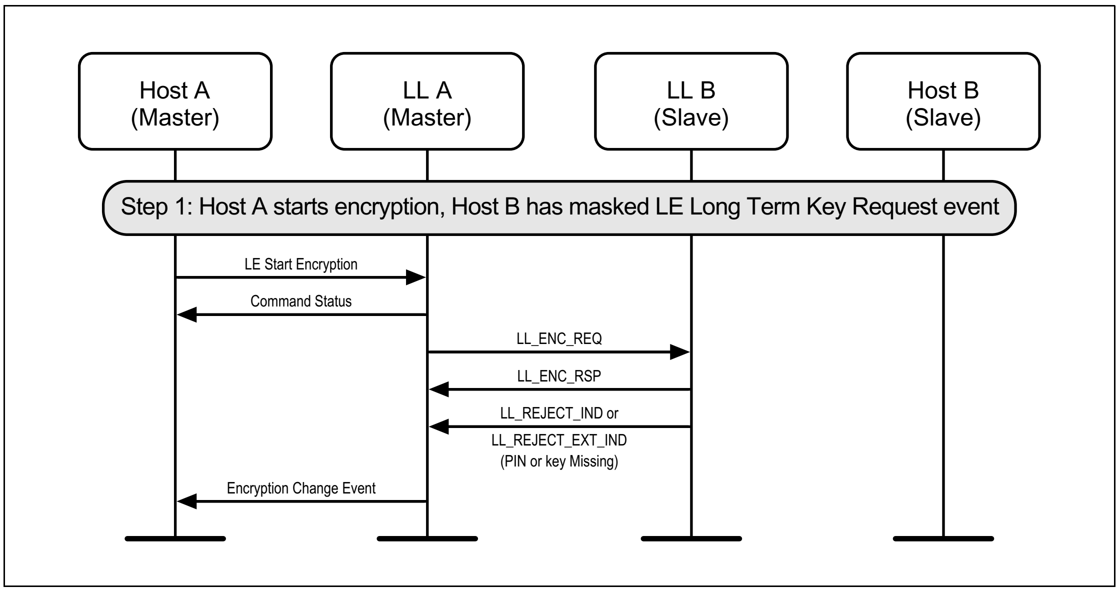

Host A (Master)
LL A (Master)
LL B (Slave)
Host B (Slave)
Step 1: Host A starts encryption, Host B has masked LE Long Term Key Request event
LE Start Encryption
Command Status
LL_ENC_REQ
LL_ENC_RSP
LL_REJECT_IND or LL_REJECT_EXT_IND (PIN or key Missing)
Encryption Change Event

Figure 6.10: 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 6.11 shows the failure case of the slave that does not support the encryption feature.

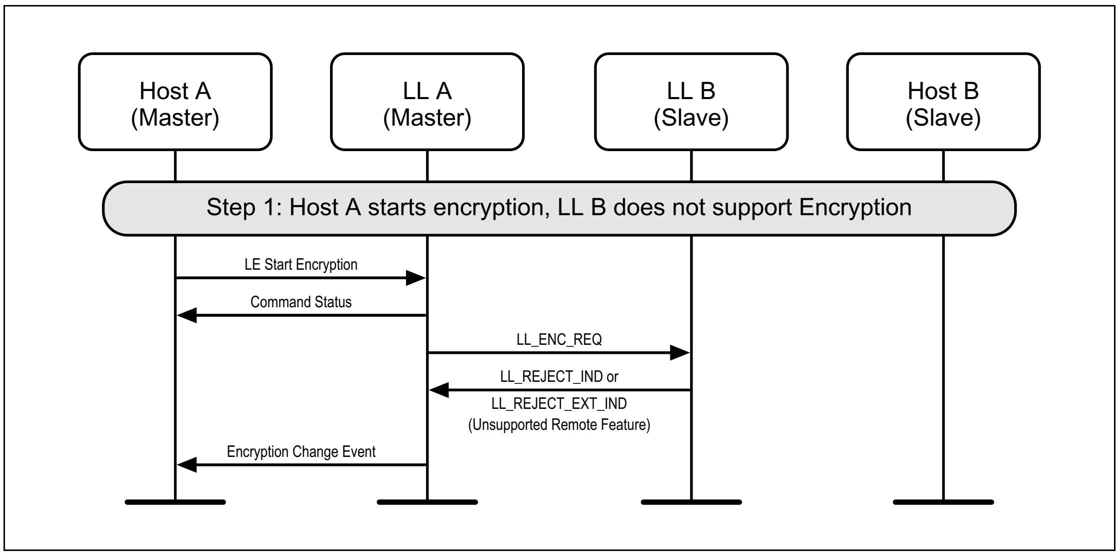

Host A (Master)
LL A (Master)
LL B (Slave)
Host B (Slave)
Step 1: Host A starts encryption, LL B does not support Encryption
LE Start Encryption
Command Status
LL_ENC_REQ
LL_REJECT_IND or LL_REJECT_EXT_IND (Unsupported Remote Feature)
Encryption Change Event

Figure 6.11: 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 6.12).

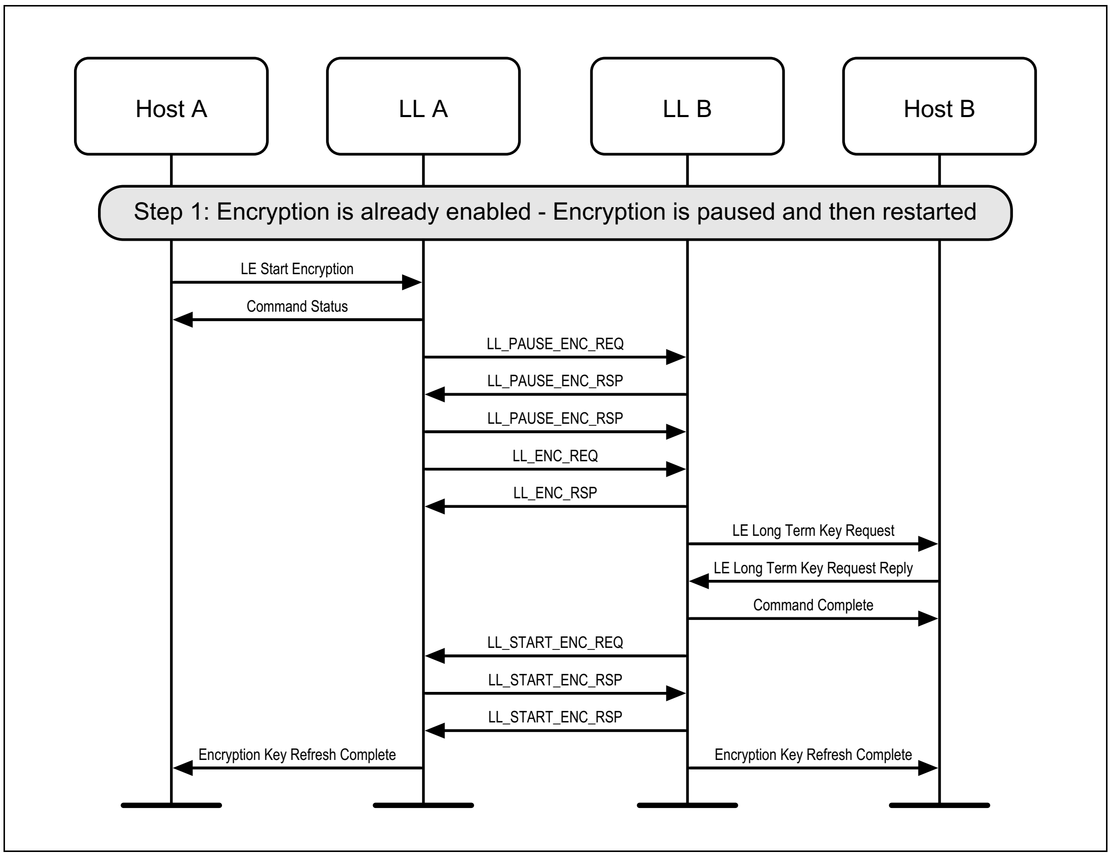


Figure 6.12: Restart Encryption

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 6.13 and Figure 6.14).

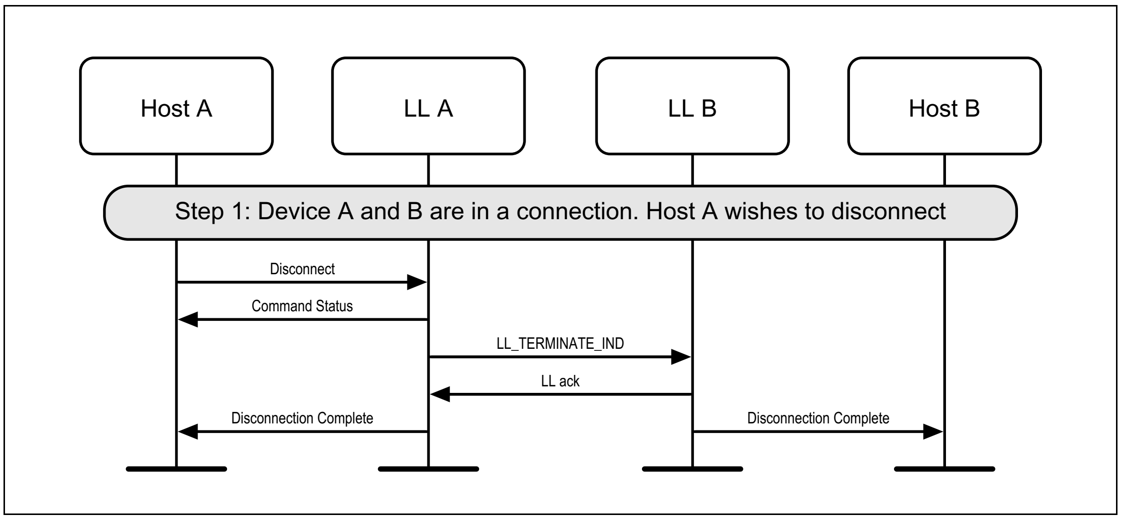


Figure 6.13: Disconnect from Master

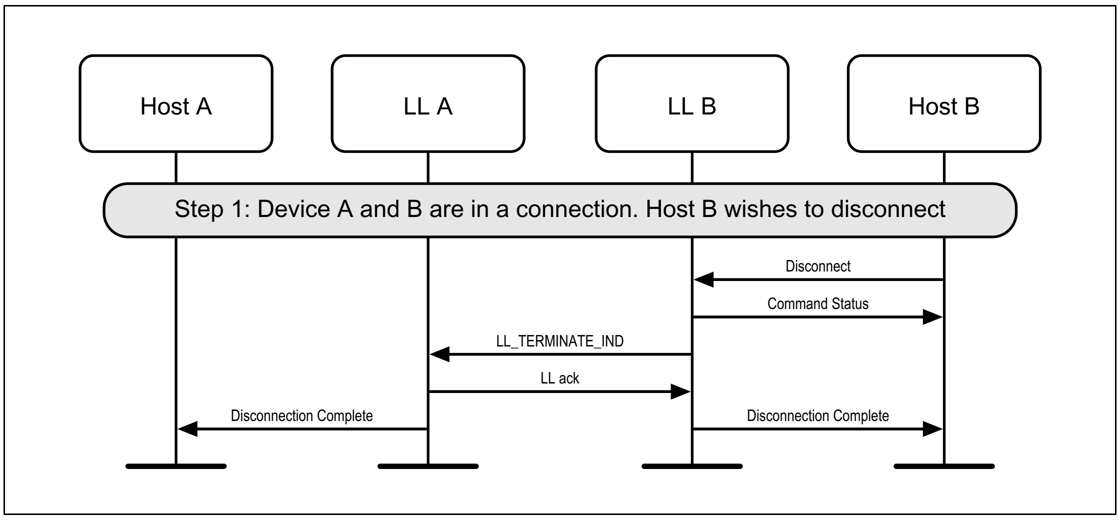


Figure 6.14: Disconnect from Slave

6.12 CONNECTION PARAMETERS REQUEST

The master or the slave of the connection may request change in connection parameters using a Link Layer Control Procedure (see Figure 6.15 to Figure 6.22).

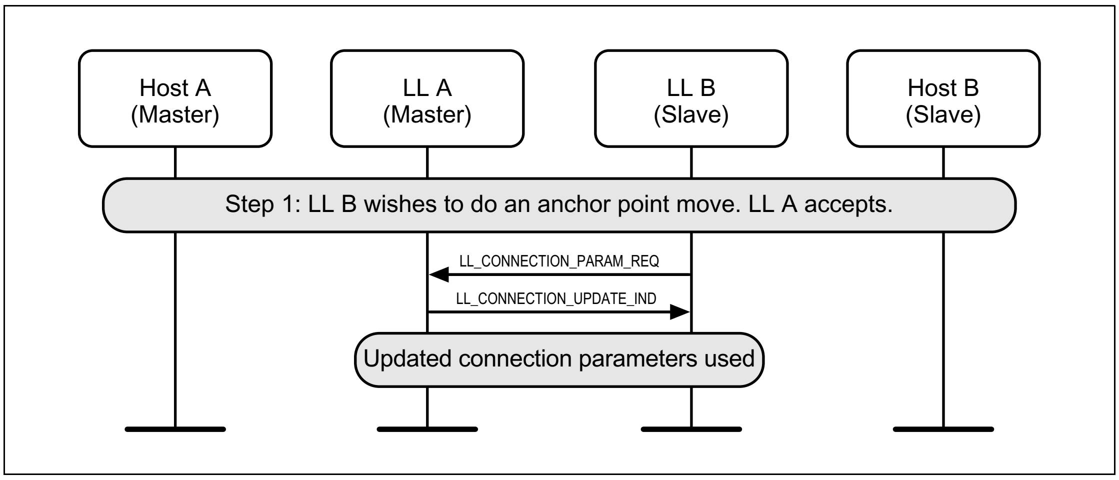


Figure 6.15: Slave-initiated Connection Parameters Request procedure – slave requests a change in anchor points, master accepts

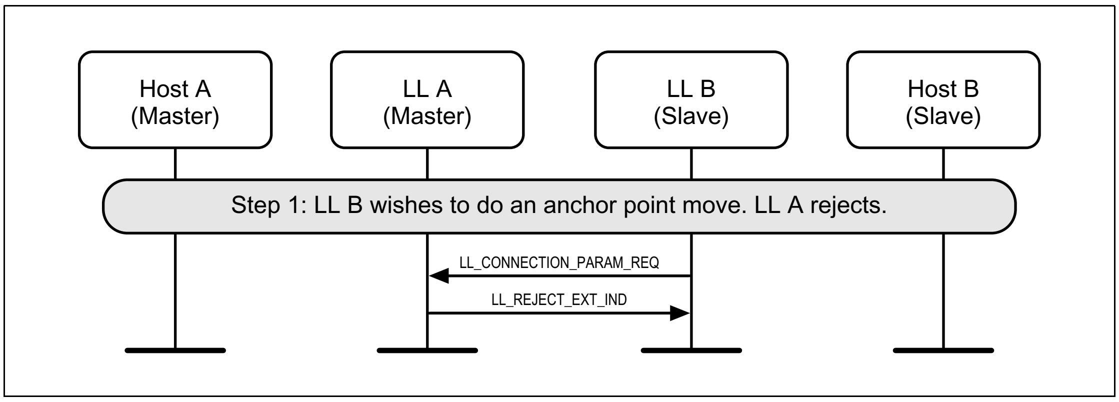


Figure 6.16: Slave-initiated Connection Parameters Request procedure – slave requests a change in anchor points, master rejects

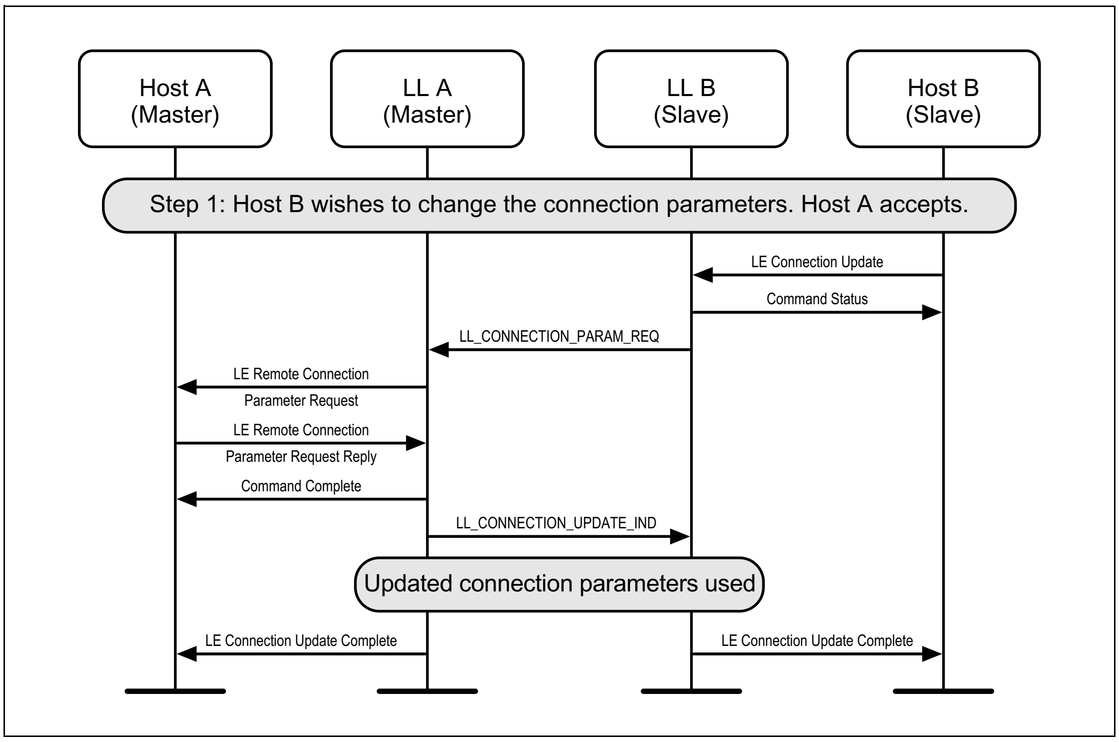

Host A (Master)
LL A (Master)
LL B (Slave)
Host B (Slave)
Step 1: Host B wishes to change the connection parameters. Host A accepts.
LE Connection Update
Command Status
LL_CONNECTION_PARAM_REQ
LE Remote Connection Parameter Request
LE Remote Connection Parameter Request Reply
Command Complete
LL_CONNECTION_UPDATE_IND
Updated connection parameters used
LE Connection Update Complete
LE Connection Update Complete

Figure 6.17: Slave-initiated Connection Parameters Request procedure – slave requests change in LE connection parameters, master's Host accepts

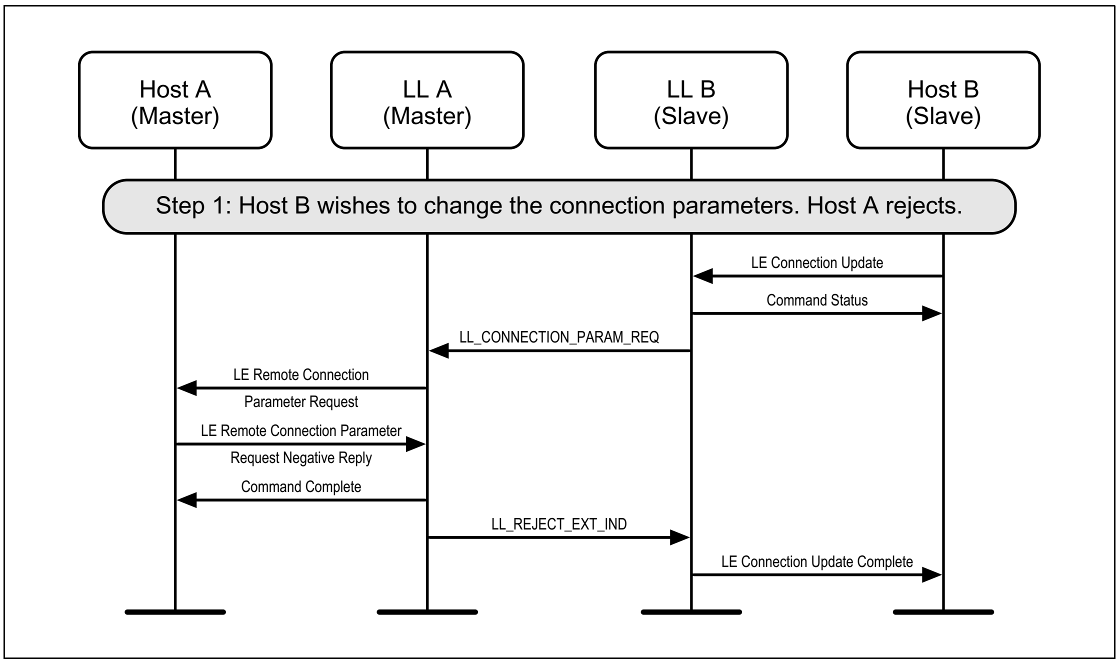

Host A (Master)
LL A (Master)
LL B (Slave)
Host B (Slave)
Step 1: Host B wishes to change the connection parameters. Host A rejects.
LE Connection Update
Command Status
LL_CONNECTION_PARAM_REQ
LE Remote Connection Parameter Request
LE Remote Connection Parameter Request Negative Reply
Command Complete
LL_REJECT_EXT_IND
LE Connection Update Complete

Figure 6.18: Slave-initiated Connection Parameters Request procedure – slave requests change in LE connection parameters, master's Host rejects

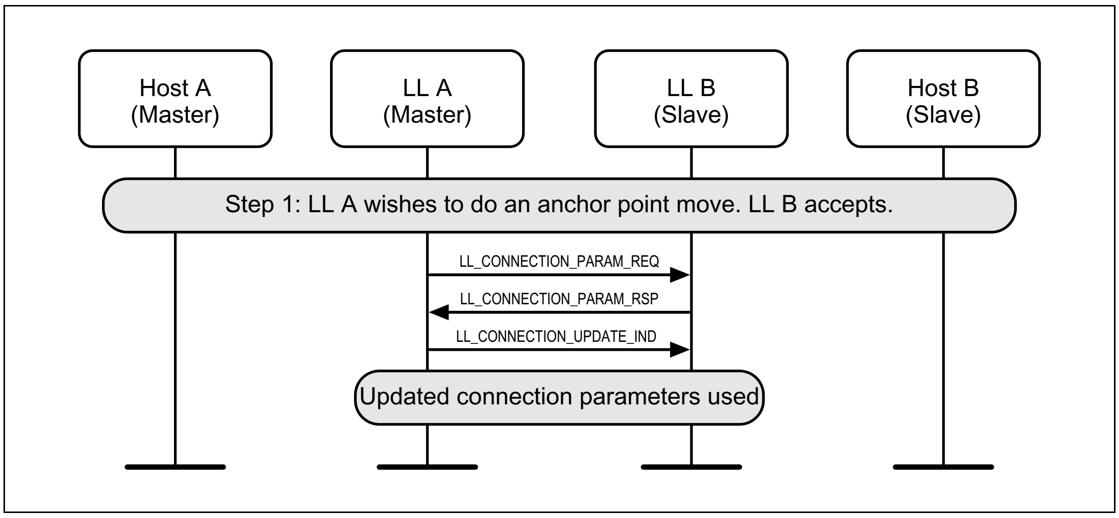


Figure 6.19: Master-initiated Connection Parameters Request procedure –master requests a change in anchor points, slave accepts

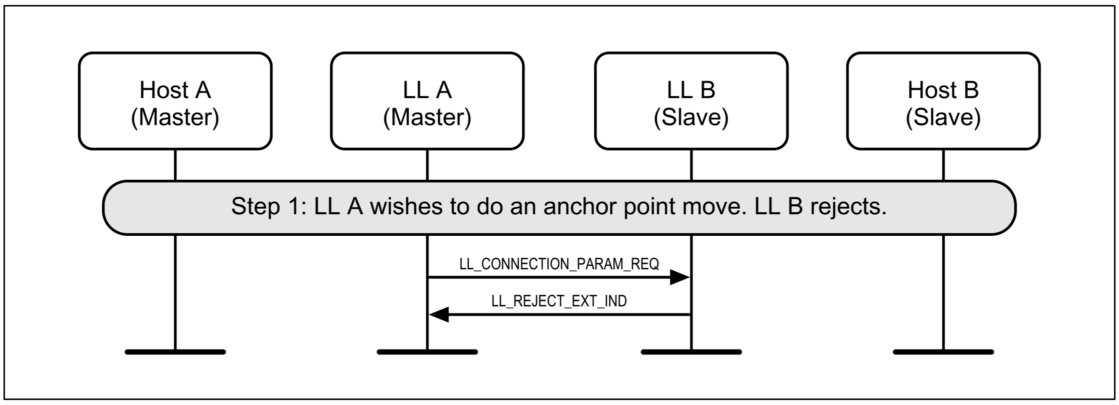


Figure 6.20: Master-initiated Connection Parameters Request procedure –master requests a change in anchor points, slave rejects

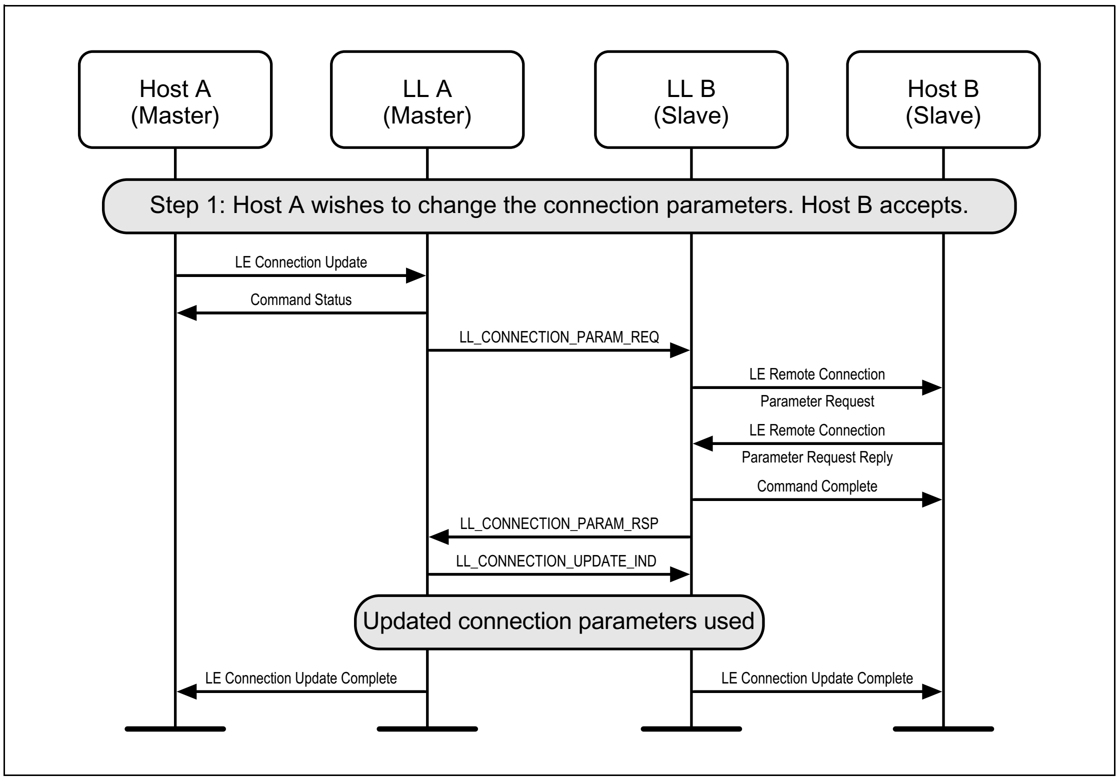

Host A (Master)
LL A (Master)
LL B (Slave)
Host B (Slave)
Step 1: Host A wishes to change the connection parameters. Host B accepts.
LE Connection Update
Command Status
LL_CONNECTION_PARAM_REQ
LE Remote Connection Parameter Request
LE Remote Connection Parameter Request Reply
Command Complete
LL_CONNECTION_PARAM_RSP
LL_CONNECTION_UPDATE_IND
Updated connection parameters used
LE Connection Update Complete
LE Connection Update Complete

Figure 6.21: Master-initiated Connection Parameters Request procedure – master requests change in LE connection parameters, slave's Host accepts

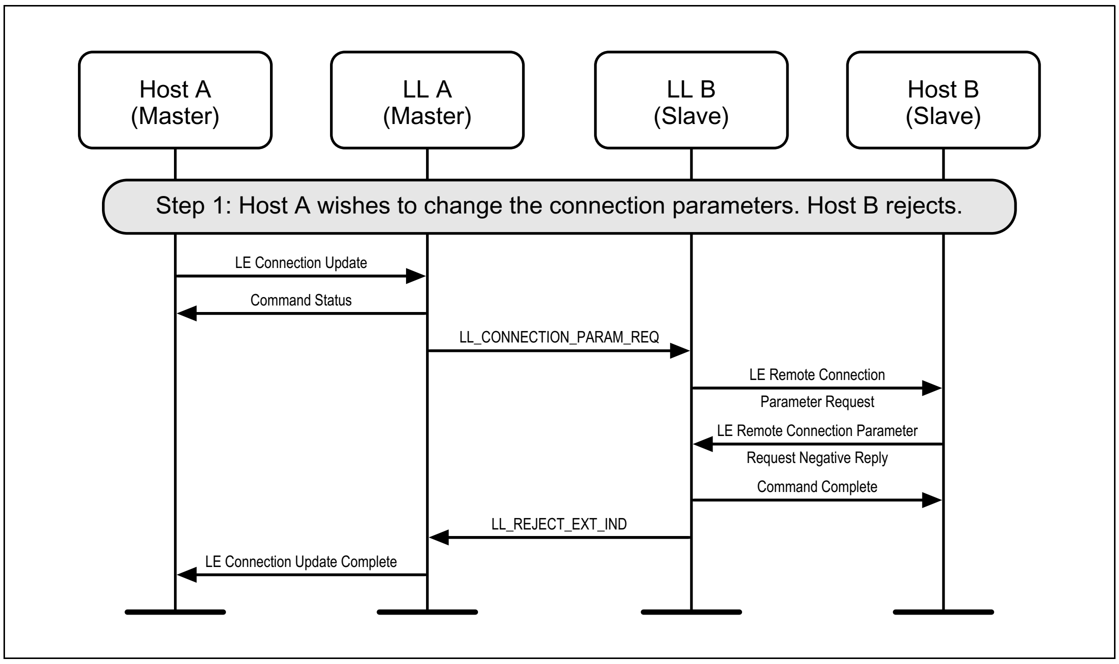

Host A (Master)
LL A (Master)
LL B (Slave)
Host B (Slave)
Step 1: Host A wishes to change the connection parameters. Host B rejects.
LE Connection Update
Command Status
LL_CONNECTION_PARAM_REQ
LE Remote Connection Parameter Request
LE Remote Connection Parameter Request Negative Reply
Command Complete
LL_REJECT_EXT_IND
LE Connection Update Complete

Figure 6.22: Master-initiated Connection Parameters Request procedure – master requests change in LE connection parameters, slave's Host rejects

6.13 LE PING

A Host may use the HCI_Write_Authenticated_Payload_Timeout command to change the maximum interval between packets containing a valid MIC that the link layer will enforce when encryption is used.

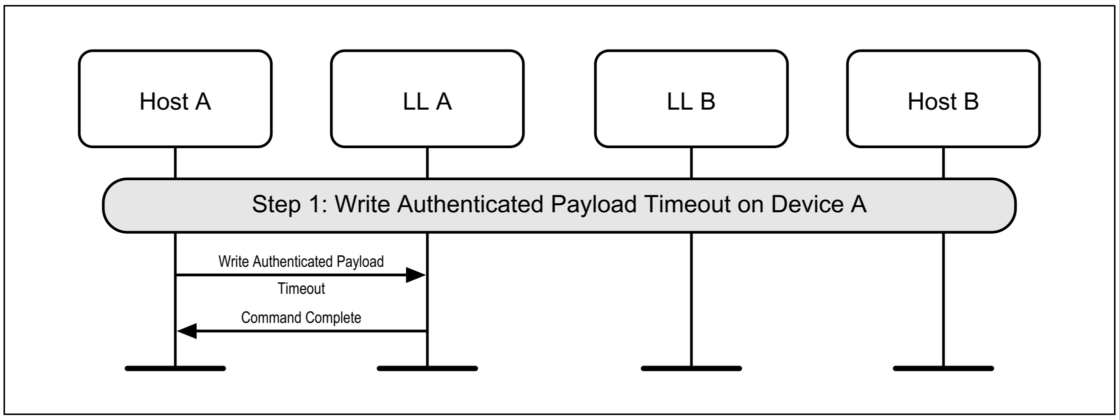

Host A
LL A
LL B
Host B
Step 1: Write Authenticated Payload Timeout on Device A
Write Authenticated Payload Timeout
Command Complete

Figure 6.23: Set LE Authenticated Payload Timeout

Either Link Layer can authenticate the remote device using the LE Ping Procedure even if the remote device does not support the LE Ping feature. This procedure can also be used for soliciting a packet from the remote device containing a valid MIC. LL A may be a master or a slave.

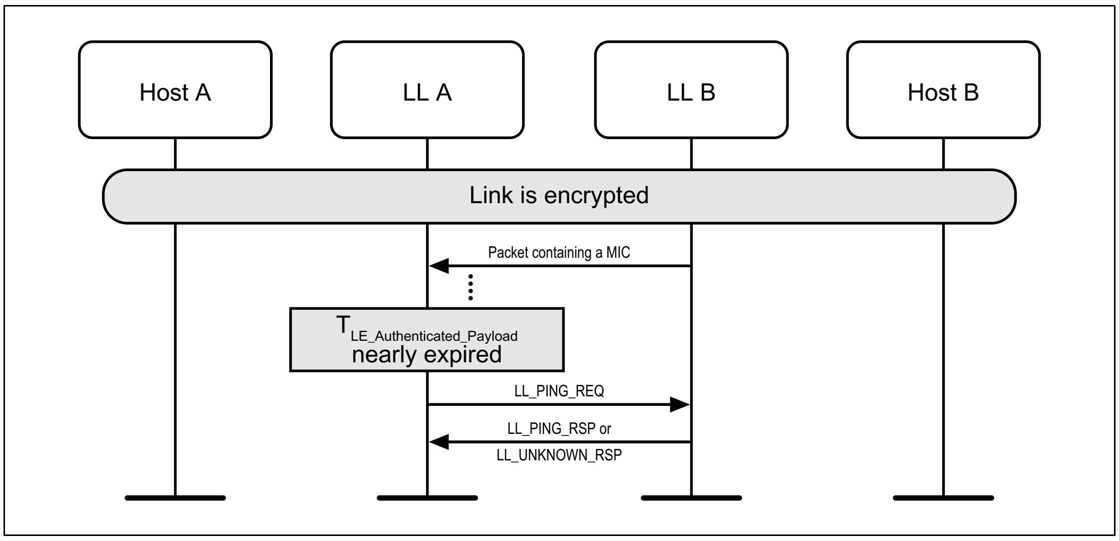


Figure 6.24: Successful LE Ping

When a packet with a valid MIC has not been received within the LE Authenticated Payload Timeout, the Host is notified that the timer has expired.

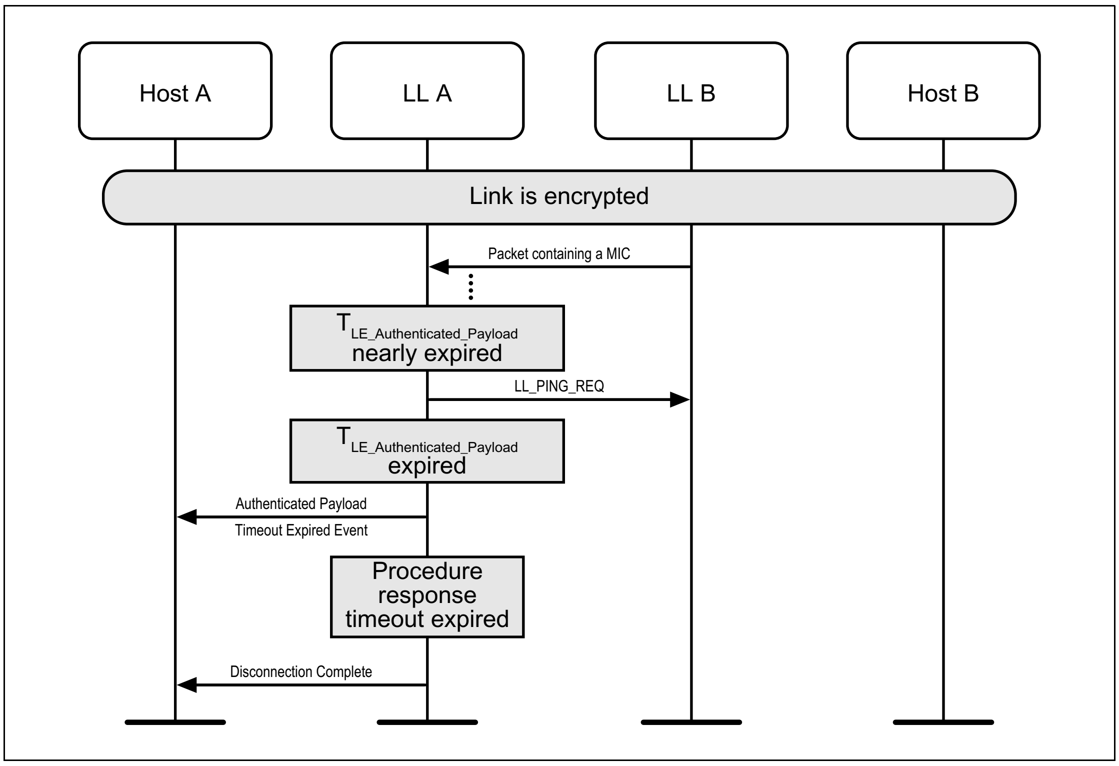

Host A
LL A
LL B
Host B
Link is encrypted
Packet containing a MIC
T LE_Authenticated_Payload nearly expired
LL_PING_REQ
T LE_Authenticated_Payload expired
Authenticated Payload Timeout Expired Event
Procedure response timeout expired
Disconnection Complete

Figure 6.25: Unsuccessful LE Ping

The $T_{LE\_Authenticated\_Payload}$ Timer gets reset when the Host sets the Authenticated Payload Timeout.

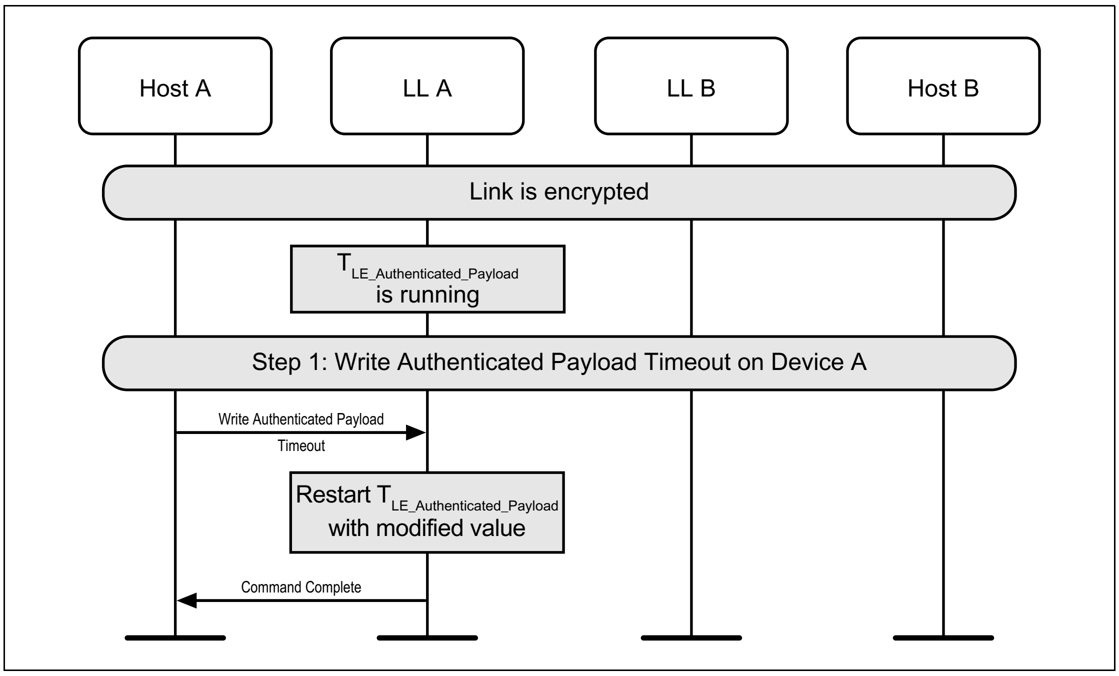

Host A
LL A
LL B
Host B
Link is encrypted
T LE_Authenticated_Payload is running
Step 1: Write Authenticated Payload Timeout on Device A
Write Authenticated Payload Timeout
Restart T LE_Authenticated_Payload with modified value
Command Complete

Figure 6.26: $T_{LE\_Authenticated\_Payload}$ Timer reset

6.14 DATA LENGTH UPDATE

Once a connection has been created, the Host may suggest maximum transmission packet size and maximum packet transmission time to be used for the connection. This may be done on either the master or the slave.

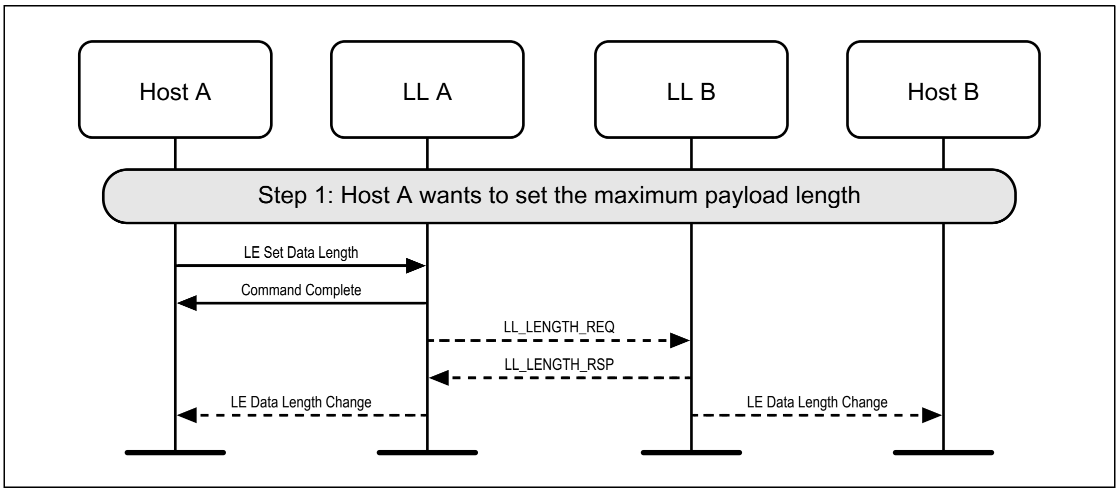


Figure 6.27: Data Length Update

6.15 PHY UPDATE

The master or slave of the connection may request a change in the PHY using a Link Layer Control Procedure (see Figure 6.28 to Figure 6.36).

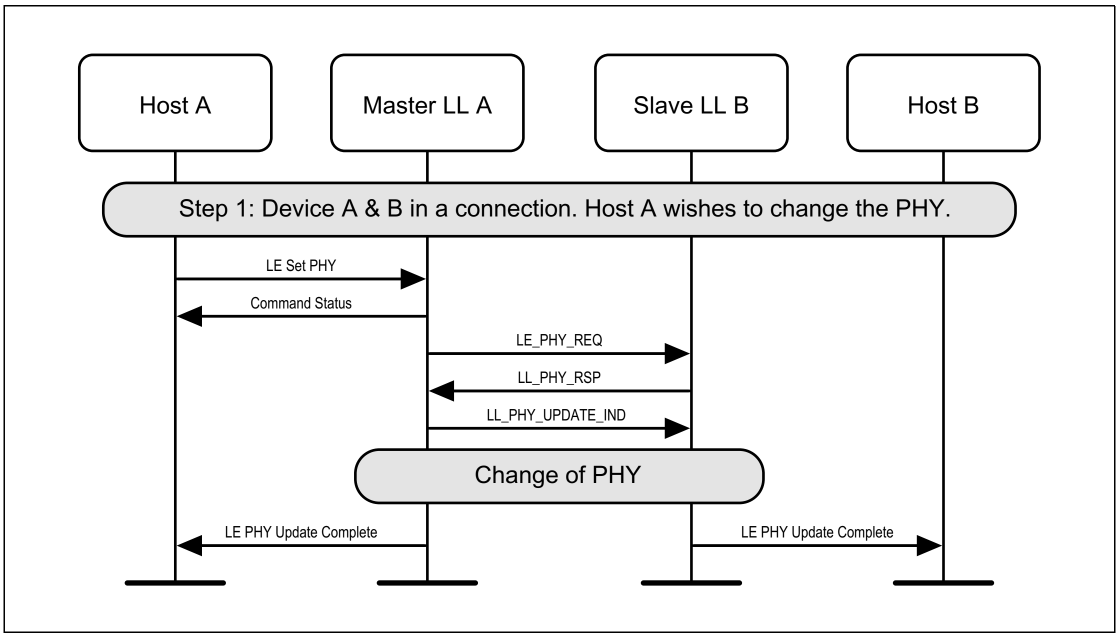


Figure 6.28: Master initiated PHY Update procedure – master requests a change of PHY, PHY changed in at least one direction

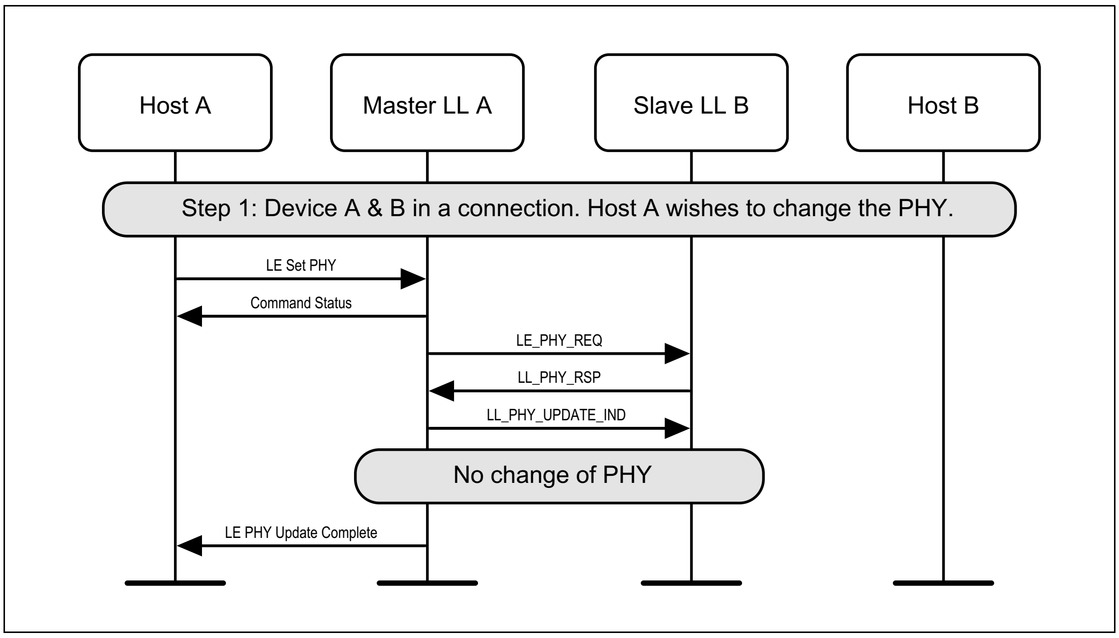


Figure 6.29: Master initiated PHY Update procedure, PHY not changed (either because slave doesn't specify PHYs that the master prefers, or because the master concludes that the current PHYs are still best)

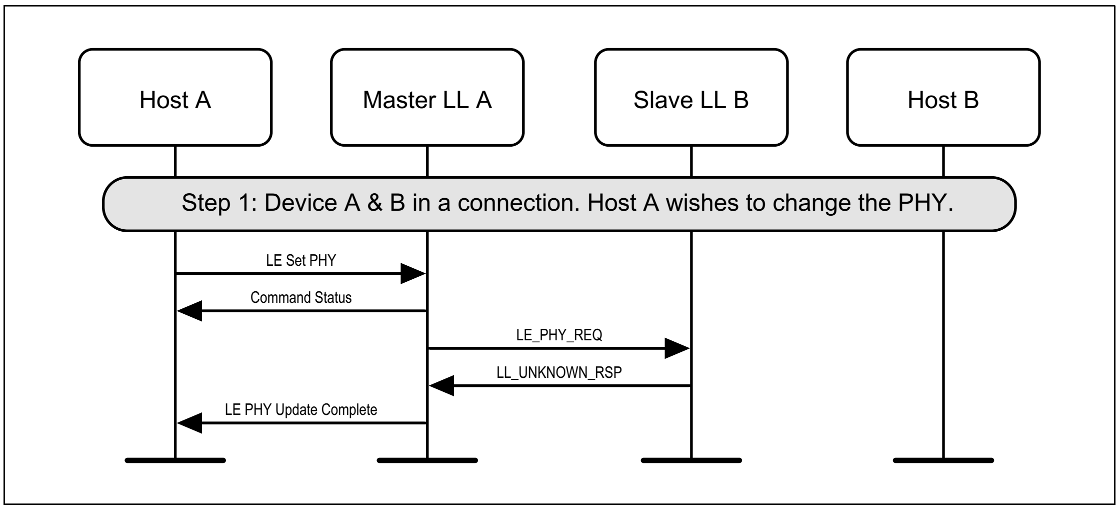


Figure 6.30: Master initiated PHY Update procedure – master requests a change of PHY, slave does not support the feature

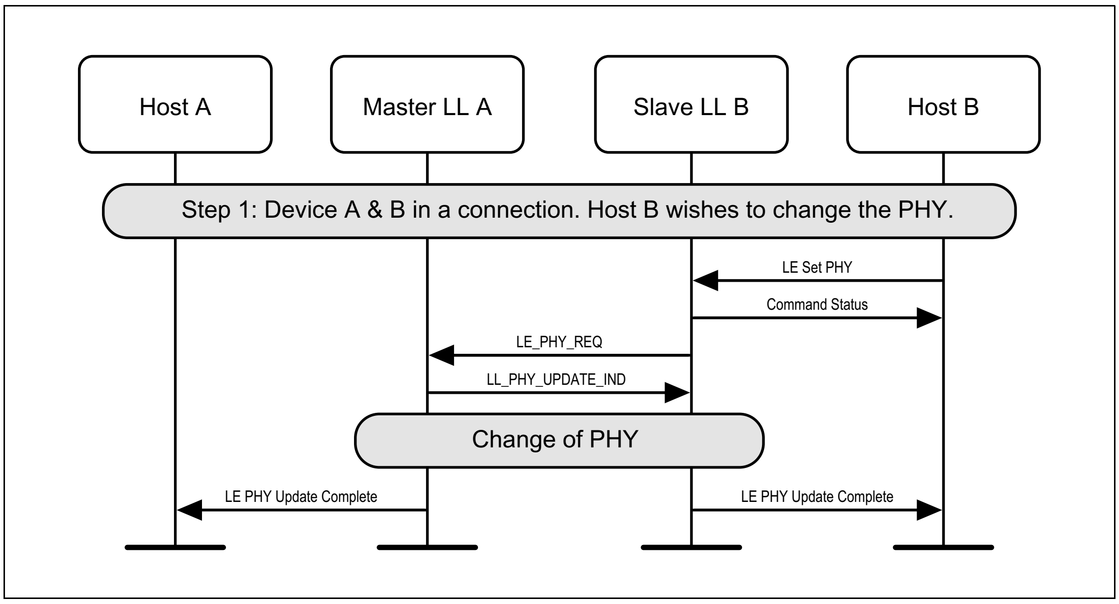


Figure 6.31: Slave initiated PHY Update procedure – slave requests a change of PHY, PHY changed in at least one direction

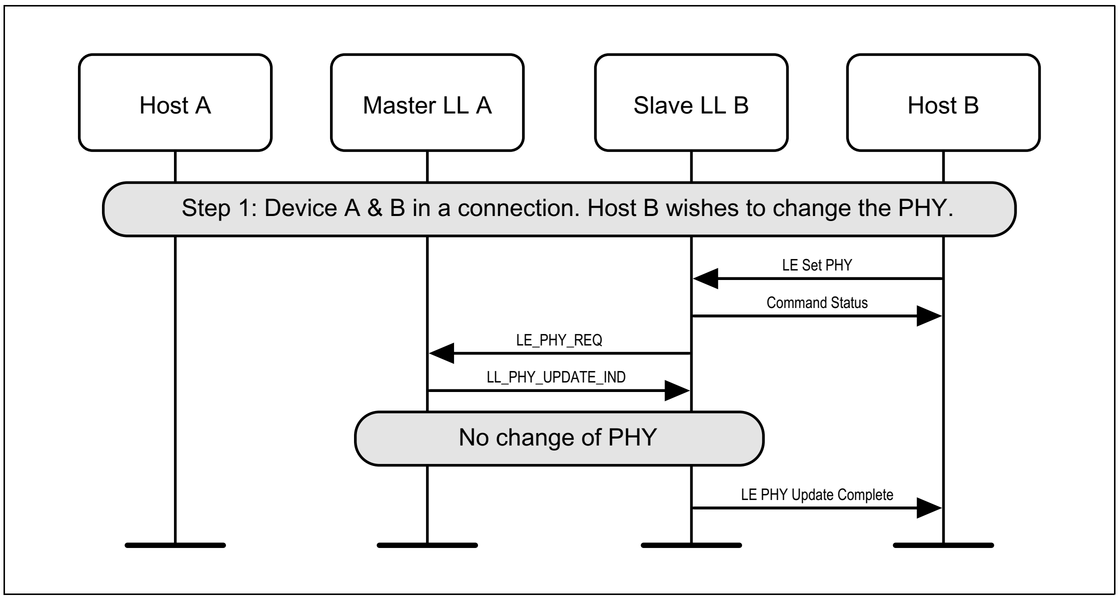


Figure 6.32: Slave initiated PHY Update procedure, PHY not changed (either because slave doesn't specify PHYs that the master prefers, or because the master concludes that the current PHYs are still best)

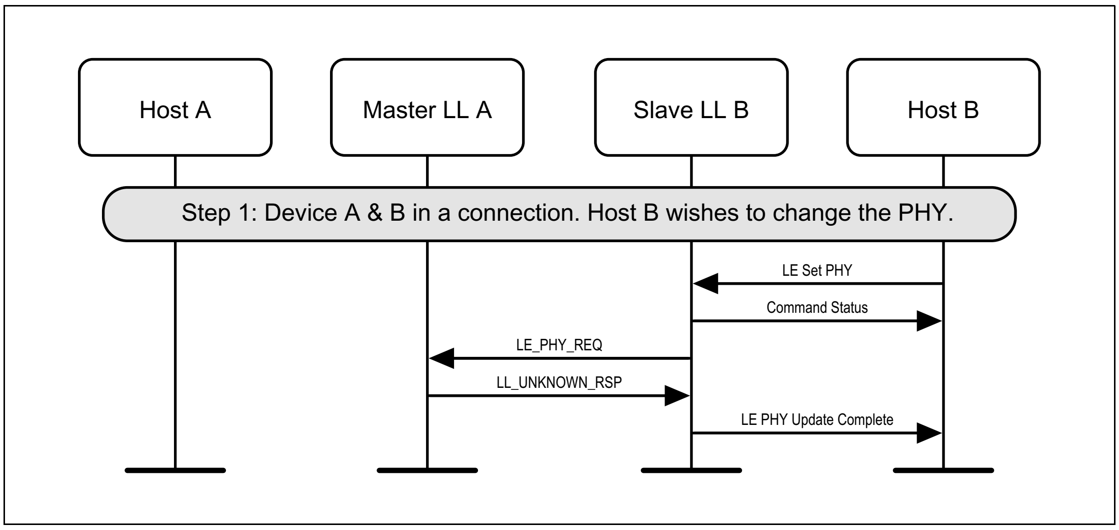


Figure 6.33: Slave initiated PHY Update procedure – slave requests a change of PHY, master does not support the feature

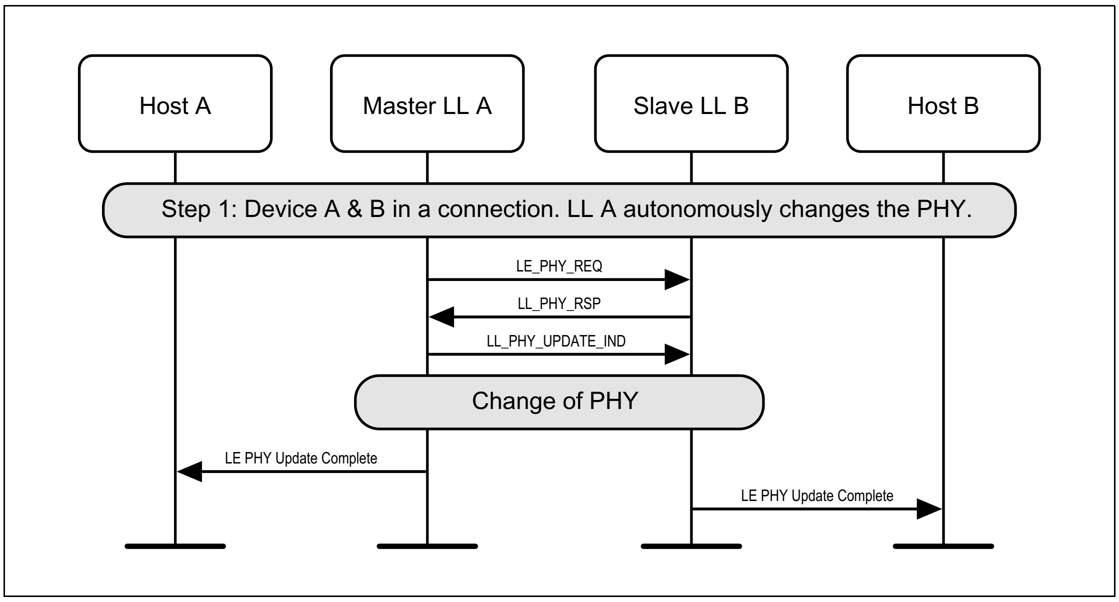


Figure 6.34: Autonomous master-initiated PHY Update procedure – master requests a change of PHY, slave accepts, PHY changed in at least one direction

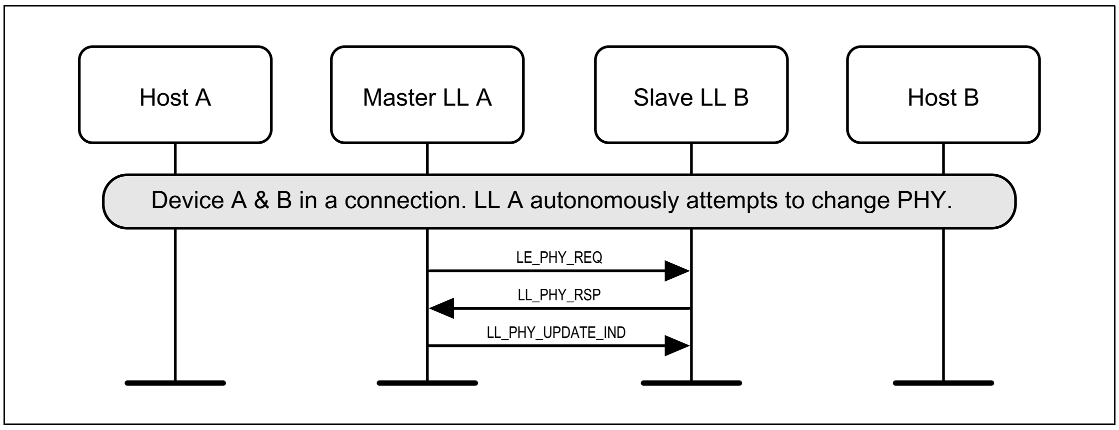

Host A
Master LL A
Slave LL B
Host B
Device A & B in a connection. LL A autonomously attempts to change PHY.
LE_PHY_REQ
LL_PHY_RSP
LL_PHY_UPDATE_IND

Figure 6.35: Autonomous master-initiated PHY Update procedure – master requests a change of PHY, PHY not changed (either because slave doesn't specify PHYs that the master prefers, or because the master concludes that the current PHYs are still best)

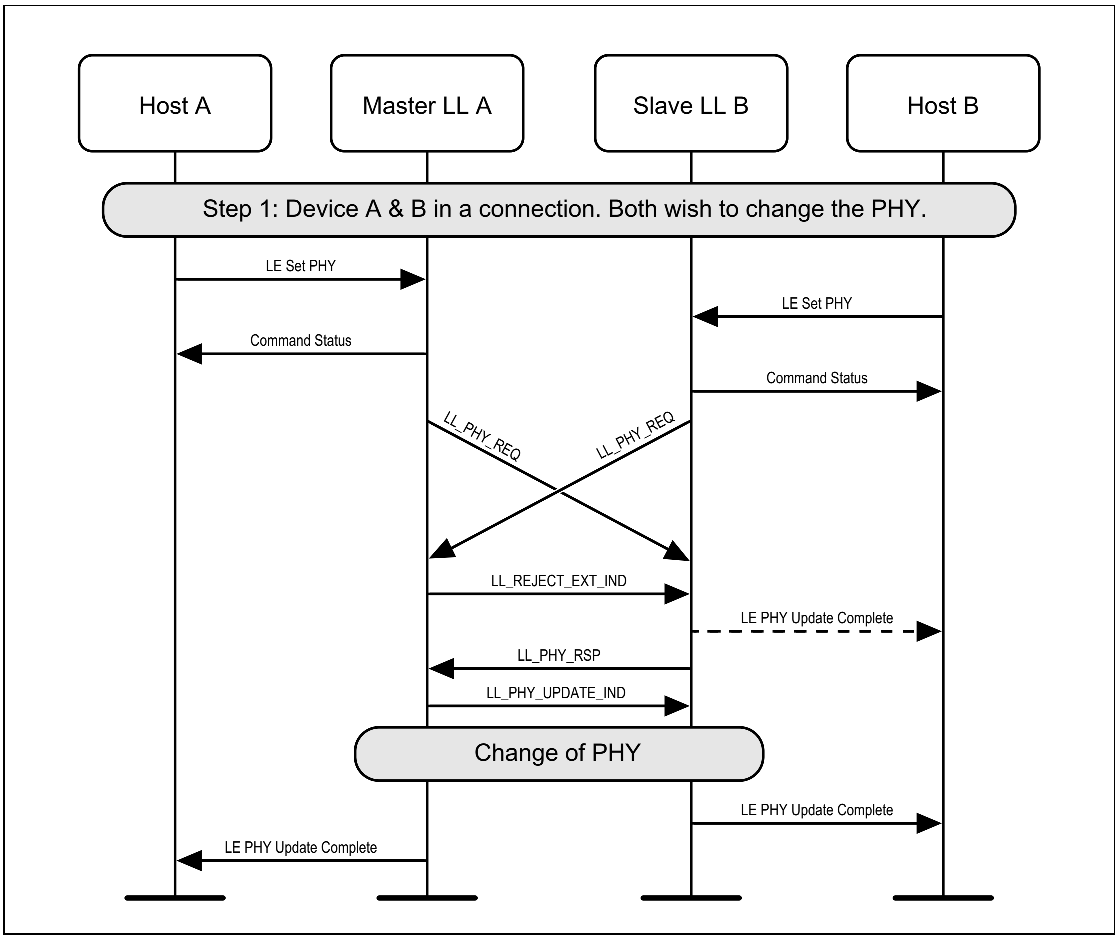


Figure 6.36: Master and slave crossover PHY Update procedure – master and slave request a change of PHY concurrently

6.16 MINIMUM NUMBER OF USED CHANNELS REQUEST

Where a slave device supports the Minimum Number of Used Channels procedure, it can request that a certain minimum number of channels be used on the indicated PHY. The example shows a successful request, resulting in a channel map update with the requested minimum number of channels used for the connection.

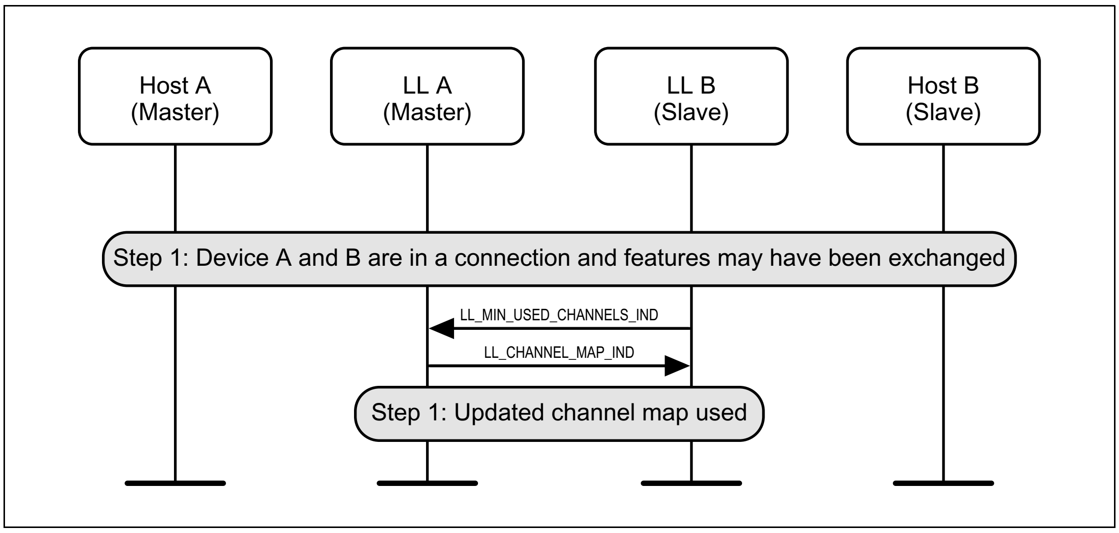

Host A
(Master)
LL A
(Master)
LL B
(Slave)
Host B
(Slave)
Step 1: Device A and B are in a connection and features may have been exchanged
LL_MIN_USED_CHANNELS_IND
LL_CHANNEL_MAP_IND
Step 1: Updated channel map used

Figure 6.37: Requesting minimum number of used channels

6.17 LL PROCEDURE COLLISION

The Link Layers of both the master and slave may initiate the same LL procedure at the same time.

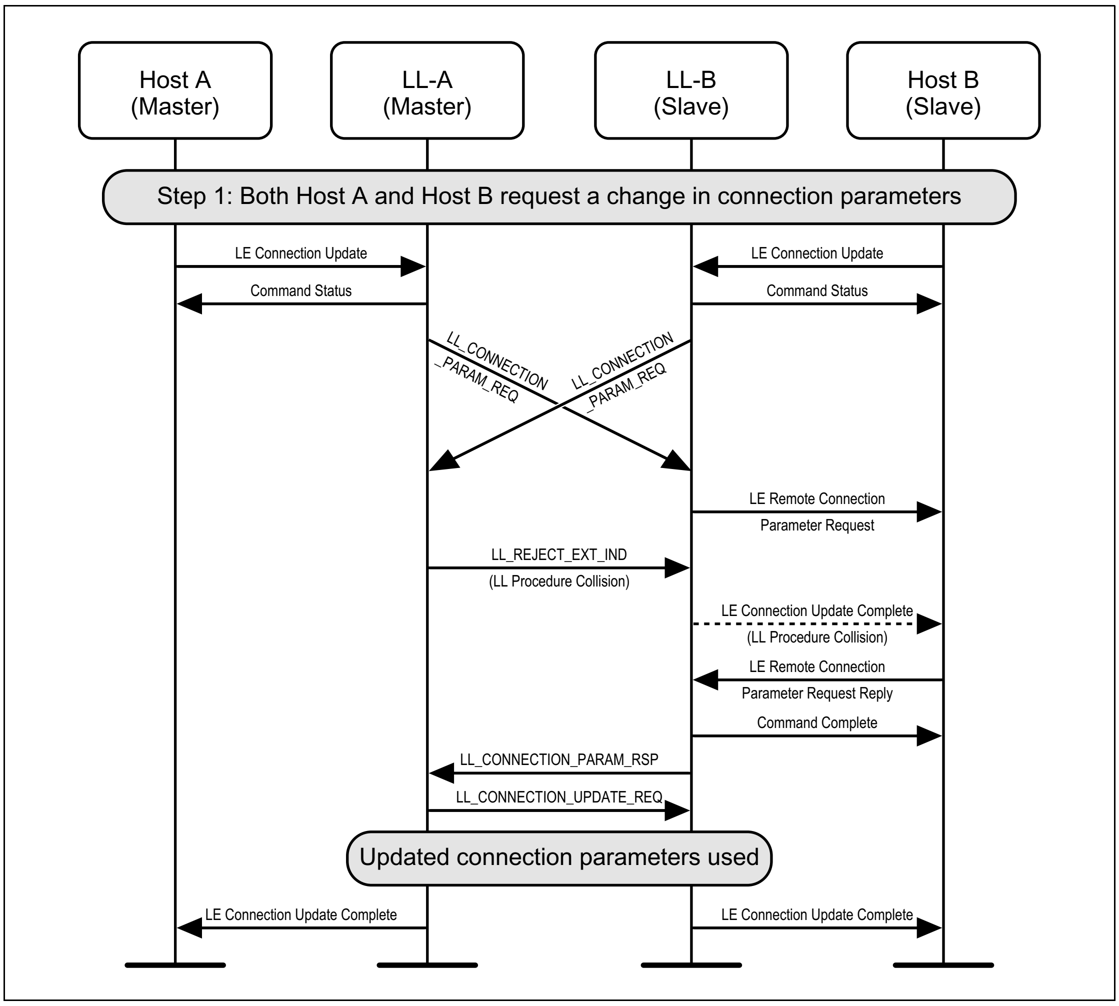

Host A (Master)
LL-A (Master)
LL-B (Slave)
Host B (Slave)
Step 1: Both Host A and Host B request a change in connection parameters
LE Connection Update
Command Status
LE Connection Update
Command Status
LL_CONNECTION _PARAM_REQ
LL_CONNECTION _PARAM_REQ
LE Remote Connection Parameter Request
LL_REJECT_EXT_IND (LL Procedure Collision)
LE Connection Update Complete (LL Procedure Collision)
LE Remote Connection Parameter Request Reply
Command Complete
LL_CONNECTION_PARAM_RSP
LL_CONNECTION_UPDATE_REQ
Updated connection parameters used
LE Connection Update Complete
LE Connection Update Complete

Figure 6.38: LL Procedure Collision

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 ([Vol 6] 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 acknowledgment 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 1 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 [Vol 6] 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

Table 2.3: Block B_1 format

Offset	Field	Size (octets)	Value	Description
2	AAD	1	variable	The data channel PDU header's first octet with NESN, SN and MD bits masked to 0
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_i 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. Subsequent blocks are 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_i format

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.

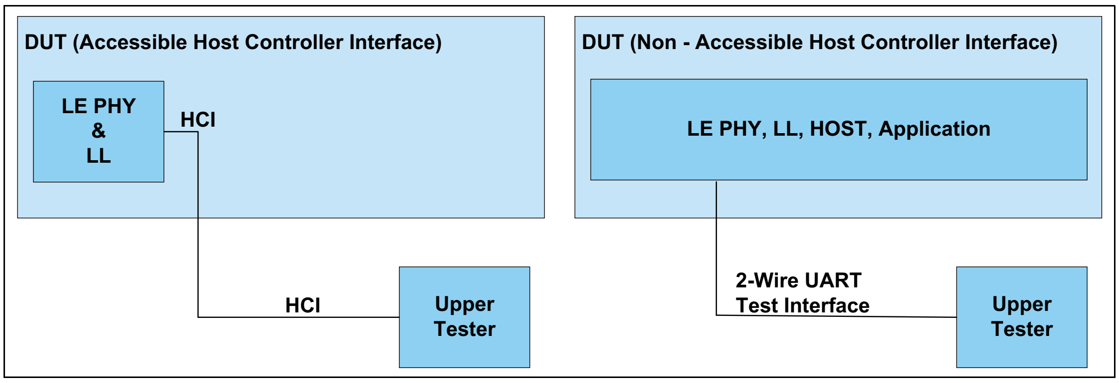

DUT (Accessible Host Controller Interface)
LE PHY & LL
HCI
HCI
Upper Tester
DUT (Non - Accessible Host Controller Interface)
LE PHY, LL, HOST, Application
2-Wire UART Test Interface
Upper Tester

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.

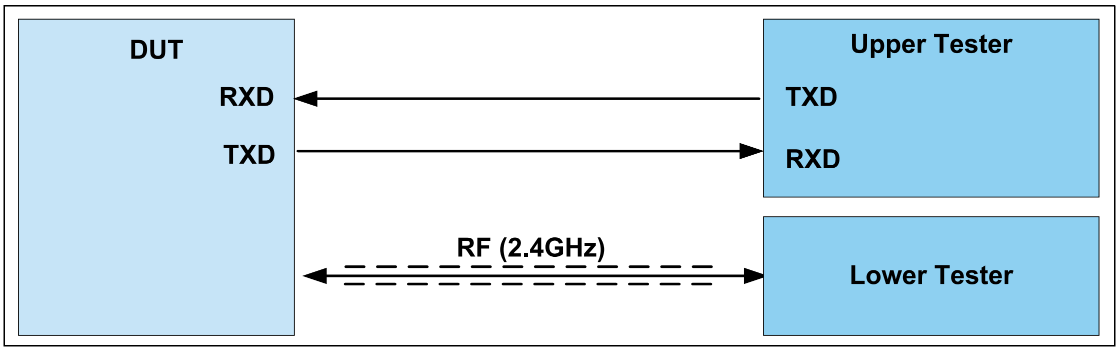


Figure 1.2: RF PHY test setup for Direct Test Mode (UART control)

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 or LE Enhanced Transmitter Test command	LE Transmitter Test
LE_RECEIVER_TEST	LE Receiver Test command or LE Enhanced 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

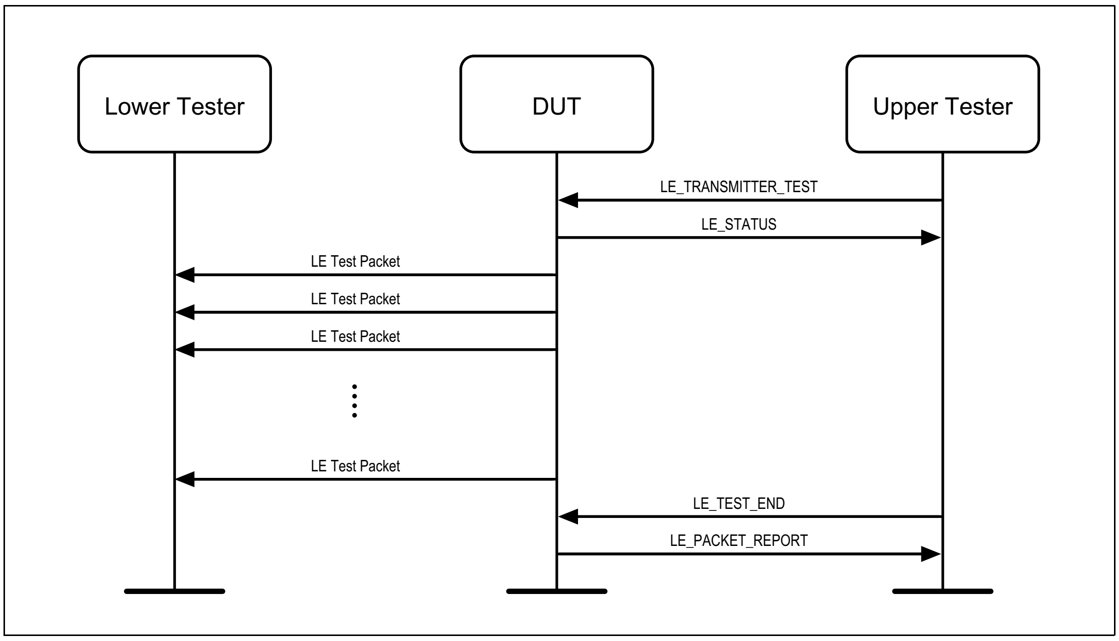


Figure 2.1: Transmitter Test MSC

Receiver Test

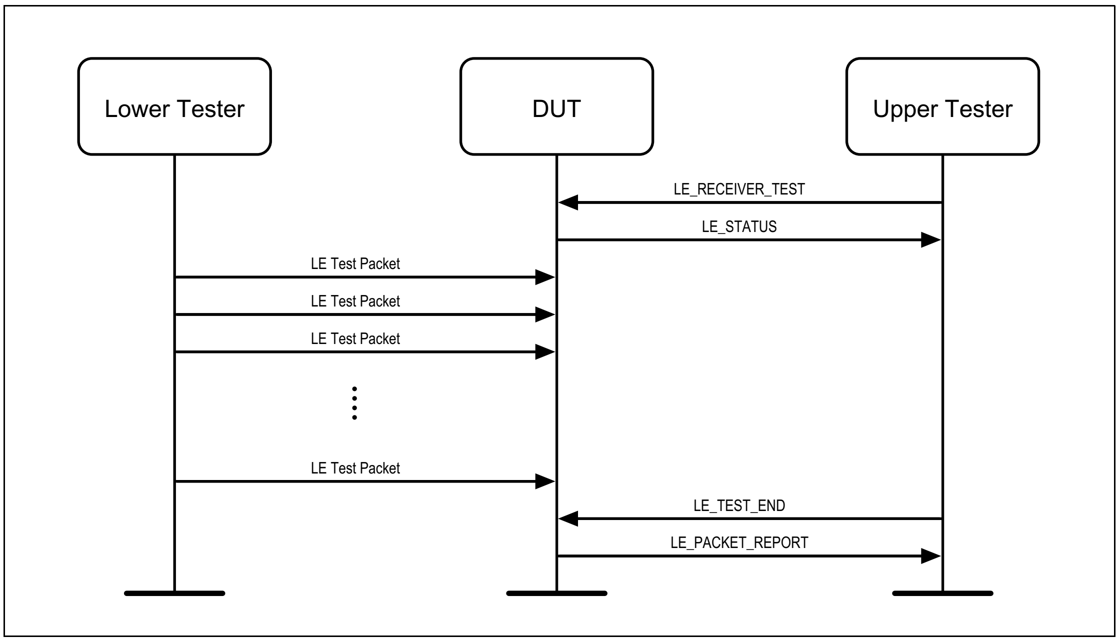


Figure 2.2: Receiver Test MSC

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 test status event or packet report 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 (i.e., a test setup command with the control argument set to 0x00) to the DUT and display an appropriate error message. For the reset command, $t_{RESPONSE}$ and $t_{TIMEOUT}$ do not apply.

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 bytes).
- The most significant bit is bit number 15.
- The least significant bit is bit number 0.
- The most significant byte is from bit 15 to 8.
- The least significant byte is from bit 7 to 0.
- Commands and Events are sent most significant byte (MSB) first, followed by the least significant byte (LSB).

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_Test_Setup	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 formats are shown in Figure 3.1 and Figure 3.2.

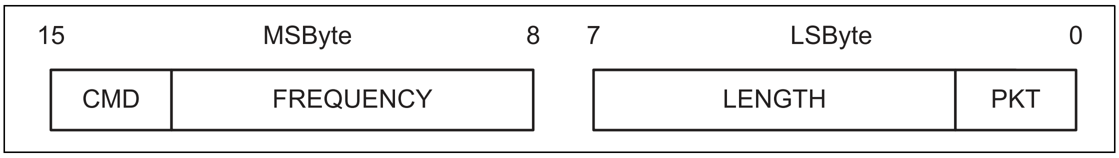


Figure 3.1: Command message format for Transmitter Test and Receiver Test commands

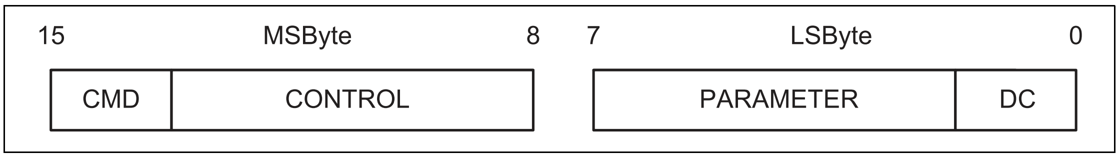


Figure 3.2: Command message format for Test Setup and Test End commands

CMD (command): *Size: 2 Bits*

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

Test Setup Command: *Size: 12 Bits*

Control (6 bits)	Parameter (6 bits)	Description
0x00	0x00	RESET; the upper 2 bits of the data length for any Transmitter or Receiver commands following are set to 00, the PHY is set to LE 1M, and the receiver assumes the transmitter has a standard modulation index
	0x01 – 0x3F	Reserved for future use
0x01	0x00 – 0x03	Set the upper 2 bits of the data length for any Transmitter or Receiver commands following (to enable a length greater than 0x3F to be used)
	0x04 – 0x3F	Reserved for future use
0x02	0x00	Reserved for future use
	0x01	PHY set to LE 1M
	0x02	PHY set to LE 2M
	0x03	PHY set to LE Coded; transmitter is to use S=8 data coding
	0x04	PHY set to LE Coded; transmitter is to use S=2 data coding
	0x05 – 0x3F	Reserved for future use
0x03	0x00	Receiver assumes transmitter has a standard modulation index
	0x01	Receiver assumes transmitter has a stable modulation index
	0x02 - 0x3F	Reserved for future use
0x04	0x00\	Read the test case supported features. The Test Status event will return the state of the test case supported features as detailed in the Test Status event (Section 3.4.1).
	Any other value	Reserved for future use

Control (6 bits)	Parameter (6 bits)	Description
0x05	0x00	Read supportedMaxTxOctets (see [Vol 6] Part B, Section 4.5.10)
	0x01	Read supportedMaxTxTime (see [Vol 6] Part B, Section 4.5.10)
	0x02	Read supportedMaxRxOctets (see [Vol 6] Part B, Section 4.5.10)
	0x03	Read supportedMaxRxTime (see [Vol 6] Part B, Section 4.5.10)
	Any other value	Reserved for future use

Test End Command: *Size: 12 Bits*

Control (6 bits)	Parameter (6 bits)	Description
0x00	0x00	Test End Command
0x00	0x01 – 0x3F	Reserved for future use
0x01 – 0x3F	0x00 – 0x3F	Reserved for future use

Transmit and receive commands:

Frequency: *Size: 6 Bits*

Value	Parameter Description
0x00 – 0x27	The frequency to be used; a value of N represents a frequency of (2N+2402) MHz (the available range is therefore even MHz values from 2402 to 2480 inclusive)
0x28 – 0x3F	Reserved for future use

Length: *Size: 6 Bits*

Value	Parameter Description
0x00 - 0x3F	The lower 6 bits of the packet length in bytes of payload data in each packet (the top two bits are set by the Test Setup command)

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	On the LE Uncoded PHYs: Vendor Specific On the LE Coded PHY: 11111111

3.4 EVENTS

There are two types of events sent by the DUT:

1. LE_Test_Status_Event
2. LE_Packet_Report_Event

The event packet format is shown in Figure 3.3. This packet format is used for both Test Status Events and Packet Report Events.

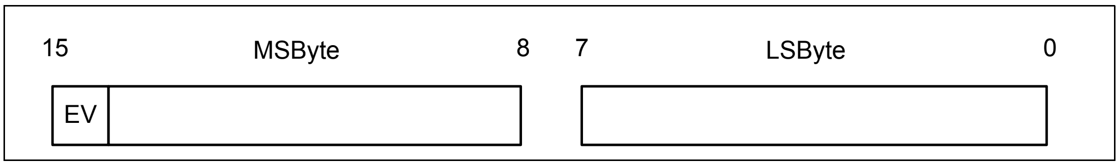


Figure 3.3: Event packet format

EV (Event): *Size: 1 Bit*

Value	**Parameter Description**
0	LE_Test_Status_Event
1	LE_Packet_Report_Event

3.4.1 LE_Test_Status_Event

The LE_Test_Status_Event packet format is as shown in Figure 3.4.

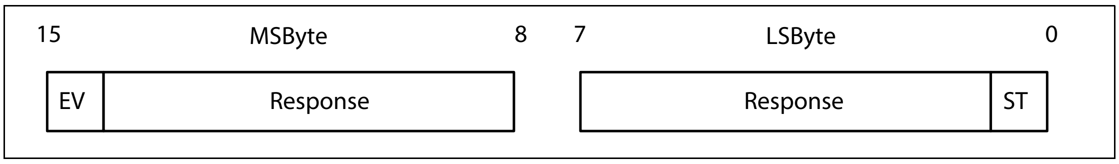


Figure 3.4: LE Test Status event

ST (status): *Size: 1 Bit*

Value	**Parameter Description**
0	Success
1	Error

Response[1] *Size: 14 Bits*

Test Setup command control parameter	**Value bits 1 to 14**[2]	
0x04	Bit 1	LE Data Packet Length Extension feature supported
	Bit 2	LE 2M PHY supported
	Bit 3	Transmitter has a Stable Modulation Index
	Bits 4 to 14	Reserved for future use
0x05	Bits 1 to 14	Maximum transmit or receive time divided by 2 or maximum number of payload octets (depending on the parameter in the original query) that the local Controller supports for transmission of a single Link Layer Data Channel PDU. Range 0x00A4-0x2148 for times or 0x001B-0x00FF for number of octets (all other values reserved for future use).
All other values		Reserved for future use

1 If the event has a status of "Error" or was generated in response to a command other than Test Setup, then this field is Reserved for future use.

2 This field is described as having bits 1 to 14 rather than 0 to 13 to avoid confusion.

3.4.2 LE_Packet_Report_Event

The LE_Packet_Report_Event packet format is shown in Figure 3.5. 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.

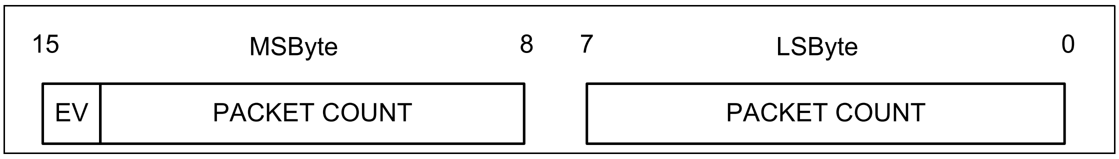


Figure 3.5: LE Packet Report event

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 byte 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
$t_{TURNAROUND}$	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

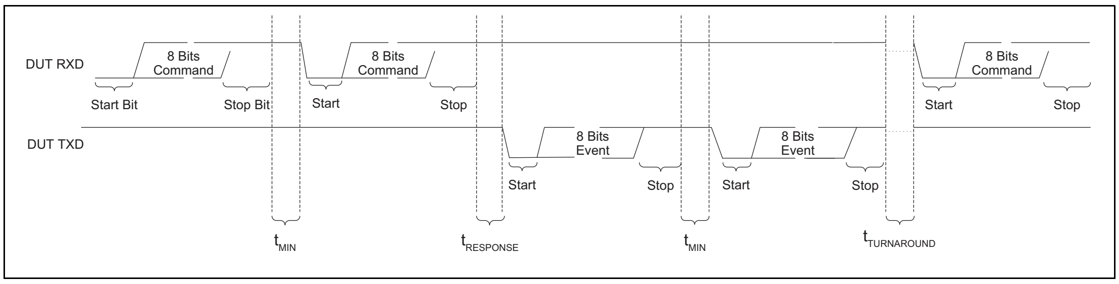


Figure 3.6: Command and event timing on 2-wire UART interface

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

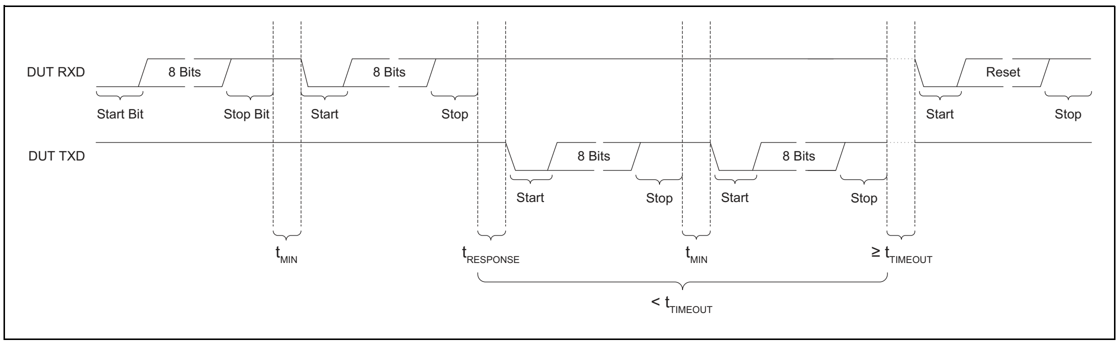


Figure 3.7: Command and event timing on 2-wire UART interface showing timeout

4 LE TEST PACKET DEFINITION

4.1 LE TEST PACKETS FORMAT

The LE Test packet format for the LE Uncoded PHYs shall be as shown in Figure 4.1. The LE Test packet format for the LE Coded PHY shall be as shown in Figure 4.2. LE test packets are required for LE RF PHY conformance testing using Direct Test Mode.

Depending on the test, the packet payload content may vary.

For the LE Uncoded PHYs, the LE test packet consists of the following fields; preamble (8 bits with the LE 1M PHY or 16 bits with the LE 2M PHY), synchronization word (32 bits), PDU header (8 bits), PDU length (8 bits), payload (0-2040 bits) and CRC (24 bits).

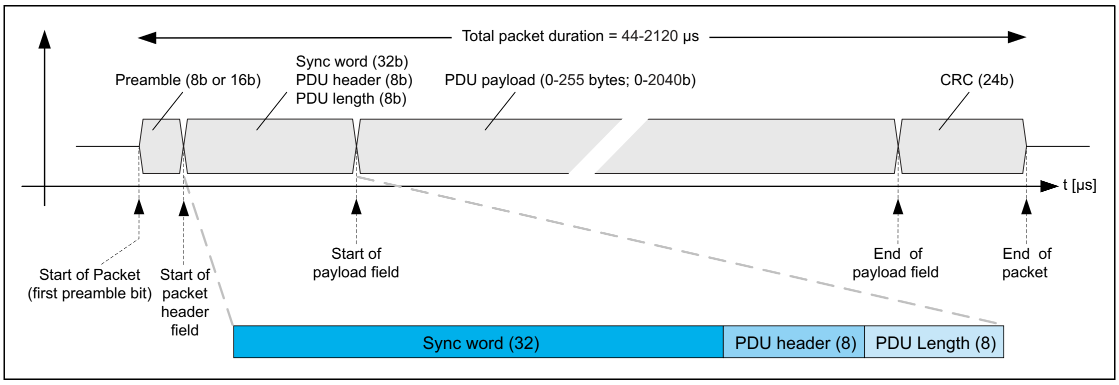


Figure 4.1: LE Test packet format for the LE Uncoded PHYs

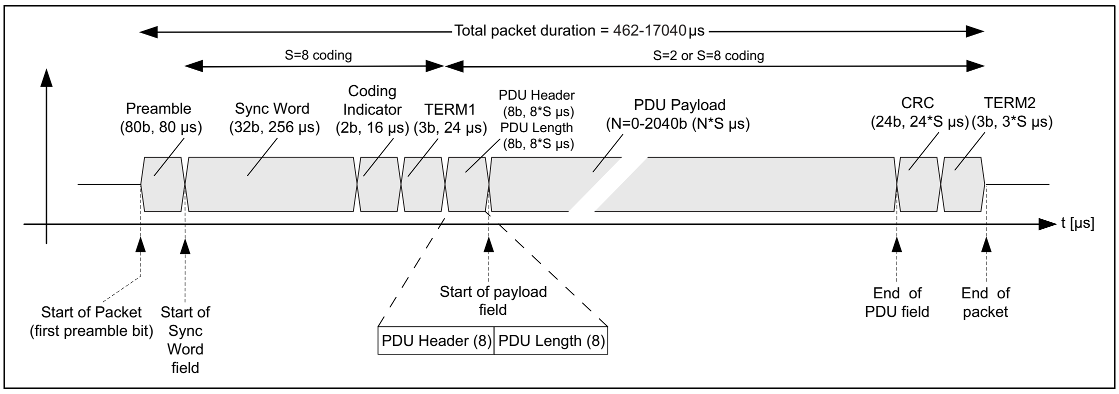


Figure 4.2: LE Test packet format for the LE Coded PHY

4.1.1 Whitening

LE test packets shall not use whitening.

4.1.2 Preamble and Synchronization Word

LE test packets shall have ‘10010100100000100110111010001110’ (in transmission order) as the synchronization word. The preamble for all LE test packets is thus ‘10101010’ (in transmission order) when the device under test is configured for the LE 1M PHY, ‘1010101010101010’ (in transmission order) if the device under test is configured for the LE 2M PHY, and the preamble described in [Vol 6] Part B, Section 2.1.1 if the device under test is configured for the LE Coded PHY.

4.1.3 CRC

The CRC shift register shall be preset with 0x555555 for every LE test packet.

4.1.4 LE Test Packet PDU

The LE test packet PDU consists of an 8-bit header, an 8-bit length field and a variable size payload. Its structure is as shown in Figure 4.3.

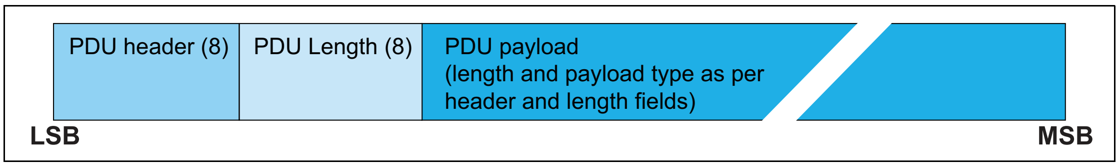


Figure 4.3: LE Test packet PDU structure

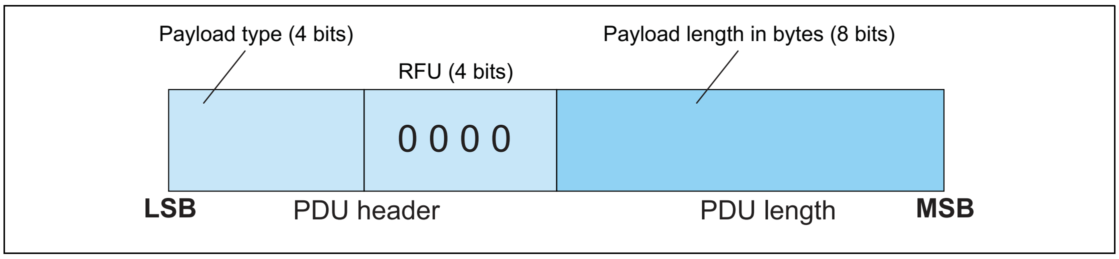


Figure 4.4: LE Test packet header and length field structure

The first four bits of the PDU header field indicate the payload content type as defined in Table 4.1. The length field expresses the payload field length in bytes.

Note: On the LE Coded PHY, this section defines the PDU contents before coding.

Payload type $b_3b_2b_1b_0$	**Payload description**
0000b	PRBS9 sequence ‘11111111100000111101…’ (in transmission order) as described in Section 4.1.5
0001b	Repeated ‘11110000’ (in transmission order) sequence as described in Section 4.1.5
0010b	Repeated ‘10101010’ (in transmission order) sequence as described in Section 4.1.5
0011b	PRBS15 sequence as described in Section 4.1.5
0100b	Repeated ‘11111111’ (in transmission order) sequence
0101b	Repeated ‘00000000’ (in transmission order) sequence
0110b	Repeated ‘00001111’ (in transmission order) sequence
0111b	Repeated ‘01010101’ (in transmission order) sequence

Table 4.1: LE test packet PDU header’s Type field encoding

Example: For LE test packets with 0x0F payload contents (‘11110000’ in transmission order) and with an LE test packet payload length of 37 bytes (296 bits), the LE test packet header and length type field will be ‘1000000010100100’ in transmission order.

4.1.5 LE Test Packet Payload Description

The LE test packet payload content alternatives required for the Bluetooth low energy RF PHY conformance tests are:

PRBS9:

A 9-bit pseudorandom binary sequence used for wanted signal payload content. The PRBS9 sequence repeats itself after the (2^9 -1 = 511) bit. The PRBS9 sequence may be generated in a nine stage shift register whose 5th and 9th stage outputs are added in a modulo-two addition stage (see Figure 4.5) and the result is fed back to the input of the first stage. The sequence begins with the first ONE of 9 consecutive ONEs (i.e. the shift register is initialized with nine ONEs).

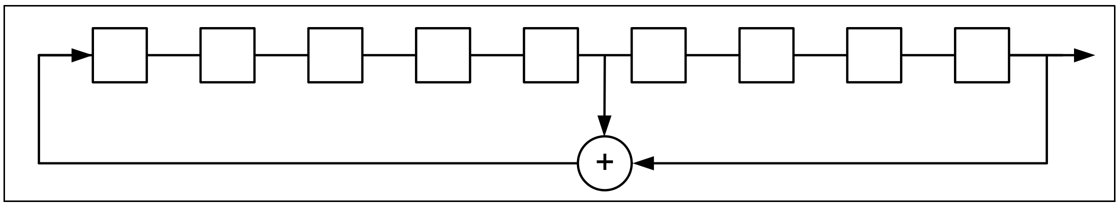


Figure 4.5: Linear feedback shift register for generation of the PRBS9 sequence

The same pseudorandom sequence of bits shall be used for each transmission (i.e. the packet is repeated).

PRBS15:

A 15-bit pseudorandom binary sequence that is used for the interfering signal and can optionally be used for wanted signal payload content. The PRBS15 sequence repeats itself after the (2^{15} -1 = 32767) bit. The PRBS15 sequence may be generated in a fifteen stage shift register whose 14th and 15th stage outputs are added in a modulo-two addition stage (See Figure 4.6) and the result is fed back to the input of the first stage. The sequence begins with the first ONE of 15 consecutive ONEs (i.e., the shift register is initialized with fifteen ONEs).

This PRBS15 definition is consistent with ITU T-REC-01 150-199605-I. SERIES O: SPECIFICATIONS OF MEASURING EQUIPMENT - Equipment for the measurement of digital and analogue/digital parameters.

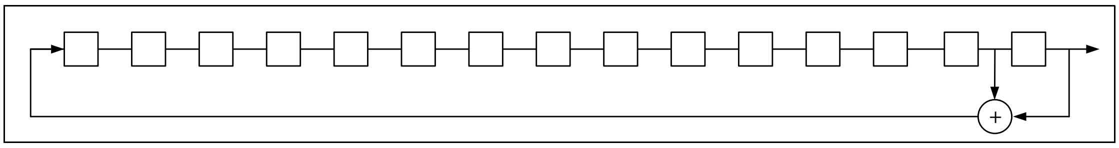


Figure 4.6: Linear feedback shift register for generation of the PRBS15 sequence

The same pseudorandom sequence of bits shall be used for each transmission (i.e. the packet is repeated).

10101010:

Repeated sequence of alternating 1’s and 0’s, starting at the first payload bit and ending at the start of the first bit in the CRC field. This pattern is used to verify the frequency deviation and the Gaussian filtering properties of the transmitter modulator.

11110000:

Repeated sequence of alternating 0’s and 1’s in groups of four (i.e. 1111000011110000…), starting at the first payload bit and ending at the start of the first bit in the CRC field. This pattern is used to verify the frequency deviation and the Gaussian filtering properties of the transmitter modulator.

4.1.6 LE Test Packet Interval

While in LE direct TX mode, LE test packets shall be transmitted from the EUT with a packet interval I(L) as defined below; see the top half of Figure 4.7 for reference.

While in LE direct RX mode, the nominal packet interval of the LE test packets transmitted from the tester is I(L), but the tester packet interval may be extended to a maximum of T(L) upon change of the dirty transmitter parameter settings and during verification of the EUT PER reporting functionality. See the bottom half of Figure 4.7 for reference.

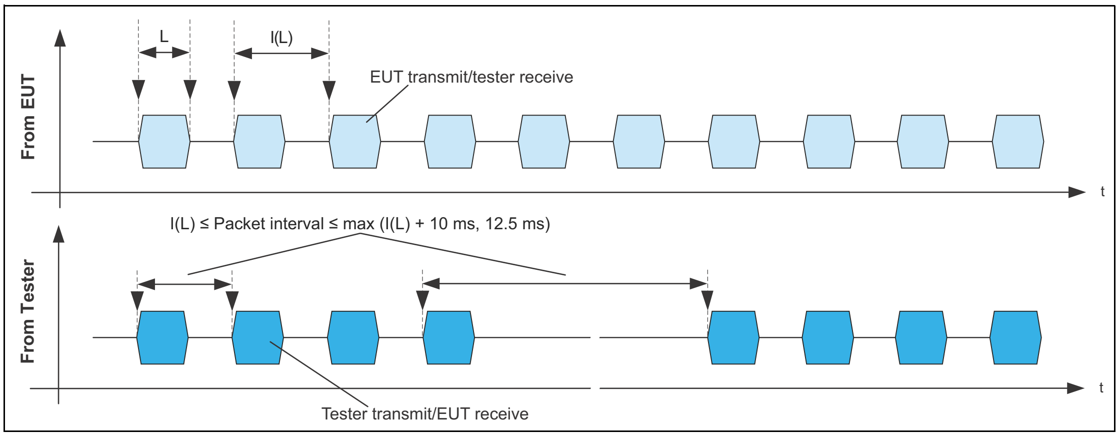

From EUT
L
I(L)
EUT transmit/tester receive
t
I(L) ≤ Packet interval ≤ max (I(L) + 10 ms, 12.5 ms)
From Tester
t
Tester transmit/EUT receive

Figure 4.7: LE Test packet interval in LE Direct Test Mode

For an LE Test Packet length of L μs, I(L) = ceil((L + 249) / 625) * 625 μs, where ceil(x) is the smallest integer greater than or equal to x, and T(L) = max (I(L) + 10 ms, 12.5 ms).