The Bluetooth® Low Energy Primer

Document Version: 1.1.0

Last updated : 17th January 2023

Contents

1. REVISION HISTORY 5

2. ABOUT THIS PAPER 6

3. INTRODUCTION 7

4. THE BLUETOOTH LE SPECIFICATIONS 8

4.1 The Bluetooth Core Specification 8

4.2 Profile Specifications 8

4.3 Service Specifications 9

4.4 Assigned Numbers 9

5. THE BLUETOOTH LE STACK 10

5.1 High-Level Architecture 10

5.2 The Layers at a Glance 12

6. THE PHYSICAL LAYER 13

6.1 Frequency Band 13

6.2 Modulation Scheme 13

6.3 PHY variants 13

6.4 Time-Division 15

6.5 Transmission Power and Receiver Sensitivity 15

6.6 Antenna Switching 15

7. THE LINK LAYER 16

7.1 Overview of the Link Layer 16

7.2 Packets 16

7.3 State Machine 17

7.4 Channel Selection 19

7.5 The Data Transport Architecture 19

7.7 The Logical Transports 20

7.7.1 LE ACL - LE Asynchronous Connection-Oriented Logical Transport 20

7.7.2 ADVB - LE Advertising Broadcast 30

7.7.3 PADVB - LE Periodic Advertising Broadcast 35

7.7.4 PAwR - LE Periodic Advertising with Responses 37

7.7.5 LE BIS and LE CIS - Isochronous Communication 41

8. THE ISOCHRONOUS ADAPTATION LAYER 49

8.1 Basics 49
8.1.1 Audio Sampling 101 49
8.1.2 Codecs and Frames 50
8.2 Framed vs Unframed 50
8.3 Fragmentation and Recombination 51
8.4 Segmentation and Reassembly 52

9. THE HOST CONTROLLER INTERFACE 54
9.1 Basics 54
9.2 The HCI Functional Specification 54
9.3 HCI Transports 54
9.3.1 UART Transport 54
9.3.2 USB Transport 55
9.3.3 Secure Digital 55
9.3.4 Three-wire UART 55
9.4 HCI Examples 55
9.4.1 Connectionless AoA/AoD 55
9.4.2 LE Path Loss Monitoring 56
9.4.3 Active Scanning 57

10. THE LOGICAL LINK CONTROL AND ADAPTATION PROTOCOL 59
10.1 Basics 59
10.2 L2CAP and Protocol Multiplexing 59
10.3 L2CAP and Flow Control 59
10.4 L2CAP Segmentation and Reassembly 60

11. THE ATTRIBUTE PROTOCOL 61
11.1 Basics 61
11.2 ATT PDUs 62
11.2.1 Commands 62
11.2.2 Requests and Responses 63
11.2.3 Notifications 63
11.2.4 Indications and Confirmations 63
11.2.5 PDU Format 64
11.2.6 Maximum Transmission Unit 64
11.3 Transactions 64
11.4 Bearers 64

11.5 Discovering Support for EATT 65

12. THE GENERIC ATTRIBUTE PROFILE 67

12.1 Basics 67

12.2 Bluetooth SIG vs Custom 68

12.3 Procedures 68

12.4 Examples 68

12.4.1 Bluetooth SIG defined attributes only 68

12.4.2 Mixture of Bluetooth SIG and Custom Attributes 69

13. THE GENERIC ACCESS PROFILE 71

13.1 Basics 71

13.2 Roles 71

13.3 Discovery 71

13.4 Connection Modes 72

13.5 Directed vs Undirected 72

13.6 Scannable vs Non-scannable 73

13.7 GAP and LE Security 73

13.8 Periodic Advertising 73

13.9 Isochronous Broadcast 74

14. THE SECURITY MANAGER PROTOCOL 75

14.1 Basics 75

14.2 Example 75

15. SECURITY IN BLUETOOTH LE 76

16. ADDITIONAL RESOURCES 77

1. Revision History

Version	Date	Author	Changes
1.0.0	22nd April 2022	Martin Woolley Bluetooth SIG	Initial version
1.0.4	6th June 2022	Martin Woolley Bluetooth SIG	Improvements to the Link Layer section: - added information that multiple link layer state machine instances are permitted. - improved language to ensure it is clear that channel classification is an optional implementation feature. - differentiated channel status reports from channel map updates. - flagged that AFH may have a different meaning in a regulatory context.
1.10	17th January 2023	Martin Woolley Bluetooth SIG	Updated to reflect Bluetooth Core Specification version 5.4, adding information about: • Periodic Advertising with Responses

2. About this paper

The Bluetooth® Low Energy Primer has been created to help technology professionals such as product designers and developers to quickly learn about Bluetooth Low Energy (LE) before consulting the formal technical specifications and delving more deeply into the subject.

There exists a substantial collection of specifications, papers and other educational resources about Bluetooth LE available from the Bluetooth SIG. A further aim of this paper is to raise awareness of their existence, their purpose and to help the reader get orientated with the subject and its supporting material.

The greater majority of Bluetooth LE products either use a combination of connectionless communication (*advertising*) and point-to-point connections to exchange data or they communicate only by broadcasting advertising packets. This resource covers the Bluetooth LE stack as it is used for products that fall into these categories. In contrast, Bluetooth mesh is not covered here. Bluetooth mesh is a specialised use of Bluetooth LE about which other information resources should be consulted

The motivation for producing this paper was the emergence of the new Bluetooth LE audio standard, *LE Audio*. Until this time, all Bluetooth audio products used the older Bluetooth technology, Bluetooth BR/EDR which is also known as Bluetooth Classic. It is possible that the advent of LE Audio means that there are numerous seasoned professionals with extensive experience of Bluetooth Classic and its use in audio products, who now find themselves needing to learn quickly about Bluetooth LE. This paper was created with such people in mind but is not exclusively for them. If you're simply new to Bluetooth LE and want to learn, you should find the paper equally useful. Don't be surprised to find an audio slant here and there though. You know the reason for this now.

It is not the aim of this paper to reproduce or cover precisely the same ground as the formal specifications or to the same depth. From time to time, brief extracts from the specifications may be included where it makes sense to do so. You should think of this paper as serving to orientate by introducing and explaining important Bluetooth LE concepts, pointing the way to other resources and specifications and hopefully, making the learning curve that little bit less steep.

3. Introduction

Bluetooth technology has been around since the year 2000. Initially created to allow two devices to exchange data wirelessly without the need for any other intermediate networking equipment it quickly found a role in products such as wireless mice and hands-free kits for cars. The latter is an audio product and audio proved to be the *killer app* for this original version of Bluetooth technology. It continued to be so for many years.

This first version of Bluetooth technology, used in those very first ever Bluetooth products is known more formally as Bluetooth BR (Basic Rate). It offered a raw data rate at the physical layer of 1 million bits per second (1 mb/s).

Later, a faster version of Bluetooth technology known as Bluetooth BR/EDR (Enhanced Data Rate) was defined. It offered a raw data rate of 2 mb/s but was still designed for use cases involving two devices exchanging data directly between them.

Bluetooth Low Energy (LE) first materialized in version 4.0 of the Bluetooth Core Specification[1]. This was a new version of Bluetooth technology which rather than replace its predecessor, Bluetooth BR/EDR, sat alongside it as an alternative with capabilities and qualities that made it perfect for a new generation of products and the ability to meet new and challenging technical and functional requirements.

Bluetooth LE supports topologies other than point to point communication between two devices with a broadcast mode which allows one device to transmit data to an unlimited number of receivers simultaneously. It is also the foundation of Bluetooth mesh networking which allows networks of tens of thousands of devices to be created, each one able to communicate with any other device in the network.

One-to-one communication between two devices is supported by both connection-oriented communication and connectionless communication. One-to-many communication is supported by connectionless broadcasting.

One of the original design goals for this new Bluetooth technology variant was to be highly efficient in its use of power. Devices which ran off small, coin-sized batteries for days or weeks or more were envisaged and that drive for energy efficiency explains many of the defining characteristics of Bluetooth LE. In particular, the design assigns asymmetrical capabilities and responsibilities to devices, seeking to ensure that devices with a relatively plentiful source of power such as a large smartphone battery do more of the heavy lifting than peer devices running on coin cell batteries. This and other design decisions like it made Bluetooth LE the low power wireless communications technology that it is and positioned it for widespread adoption in a multitude of types of product in the years that would follow.

[1] The Bluetooth specifications are introduced in section 4

4. The Bluetooth LE Specifications

A deep and thorough understanding of Bluetooth LE requires intimate familiarity with the applicable specifications. The architecture, procedures and protocols of Bluetooth LE are defined in full by one key specification called the Bluetooth Core Specification. How products use Bluetooth such that they are interoperable is covered by collections of specifications of two special types known as *profiles* and *services*. Figure 1 illustrates the Bluetooth LE specification types and their relationships.

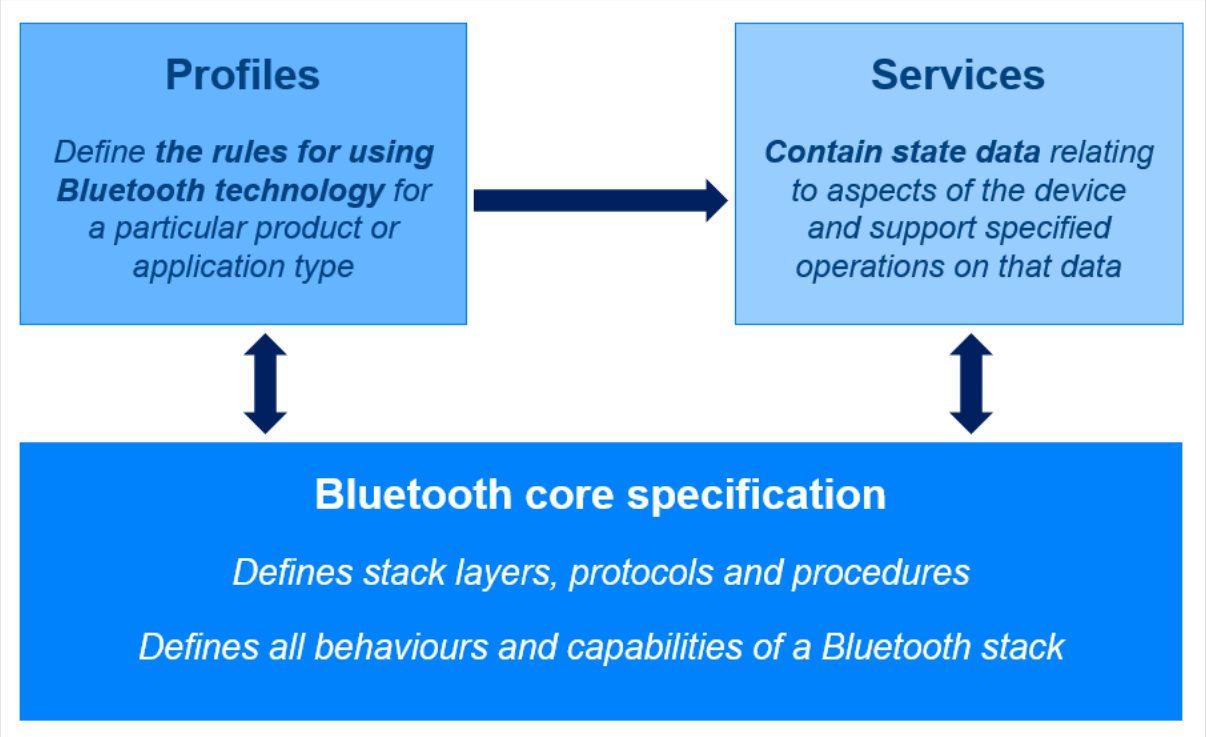


Figure 1 - The Bluetooth LE specifications

4.1 The Bluetooth Core Specification

The Bluetooth Core Specification is the primary specification for both Bluetooth LE and Bluetooth Classic. It defines the architecture of the technology and its layers, it describes and defines its key features and it defines the formal procedures underpinning important operations and the protocols with which devices can communicate at a given layer of the stack. It is a necessarily large specification.

The Bluetooth Core Specification defines how Bluetooth technology works and the requirements for developers when implementing a Bluetooth stack or one or more of its features.

4.2 Profile Specifications

When two Bluetooth LE devices are communicating over a connection, it is usually the case that a client / server relationship has been formed. Servers contain state data and clients use that data in some way.

Consider a Bluetooth key fob whose purpose is to help you find your keys after you've put them down somewhere and temporarily lost them. A smartwatch might act as the client device and the Bluetooth key fob would act as the server. Pressing a button on the smartwatch's display could

cause the key fob's state to be changed and it to make a loud noise in response to this change so that you can find your keys again.

Profile specifications define the roles that related devices (like the smartwatch and key fob) assume and in particular, define the behaviour of the client device and the data on the connected server that it should work with.

In the key finder example, the behaviour of the smartwatch or some other device assuming the same role is defined in the Find Me Profile specification.

4.3 Service Specifications

State data on servers resides in formally defined data items known as *characteristics* and *descriptors*[2]. Characteristics and descriptors are grouped inside constructs known as *services*. Services provide a context within which to assign meaning and behaviours to the characteristics and descriptors that they contain.

A service specification defines a single service along with the characteristics and descriptors that it contains. The behaviours to be exhibited by the device hosting the service in response to various conditions and state data values are defined in the service specification.

A service specification can be thought of as defining one aspect of a server device's behaviour.

In the smartwatch and key fob example, the key fob, is acting as a server and implements the Immediate Alert Service.

4.4 Assigned Numbers

Various aspects of Bluetooth LE make use of unique identifiers. For example, all services, characteristics and descriptors have a universally unique identifier (UUID) which identifies the *type* of the service, characteristic or descriptor to which it relates rather than a particular instance on a particular device. A company may be identified by a unique company identifier, something which is required by certain profiles.

Identifiers allocated by the Bluetooth SIG are known as *assigned numbers* and a full list of such identifiers can be obtained from the Assigned Numbers page on the Bluetooth SIG website.

[2] Services, characteristics and descriptors are explained in the Generic Attribute Profile section

5. The Bluetooth LE Stack

5.1 High-Level Architecture

The Bluetooth LE stack consists of a number of layers and functional modules, some of which are mandatory and some of which are optional. These parts of the stack are distributed across two major architectural blocks known as the *host* and the *controller* and a standard logical interface defines a way in which these two components may communicate.

The host is often something like an operating system. The controller is often a system on a chip. This is not necessarily the case however and the Bluetooth specifications do not mandate any such implementation details. What's important is that the host and controller act as separate logical containers in the architecture which *may* be implemented in physically separate components in some way, with a standard interface defined for communication between them. This allows a Bluetooth system to consist of host and controller components from different manufacturers.

Figure 2 illustrates the Bluetooth LE stack, its layers and their distribution across the host and controller components.

The Host Controller Interface (HCI) indicates the logical interface between them but is not a physical component as such. HCI can be implemented in a number of different ways in terms of the underlying physical transport but the logical or functional interface is always the same.

LC3 is the Low Complexity Communication Codec, the default audio codec used with Bluetooth LE Audio. It is not part of the standard Bluetooth LE stack as such but will always be found in LE Audio products, with the LC3 component either implemented in the host or in the controller as shown.

Figure 3 illustrates the standard OSI reference model for communications systems. It should be noted that the Bluetooth LE stack spans all layers of the OSI reference model in contrast to many other wireless systems which span a subset of the OSI layers only, such as the physical and data link layers. One advantage that Bluetooth technology has through being a full stack communication system is that there are no external dependencies on other standards bodies. Such dependencies can constrain the evolution of a technology.

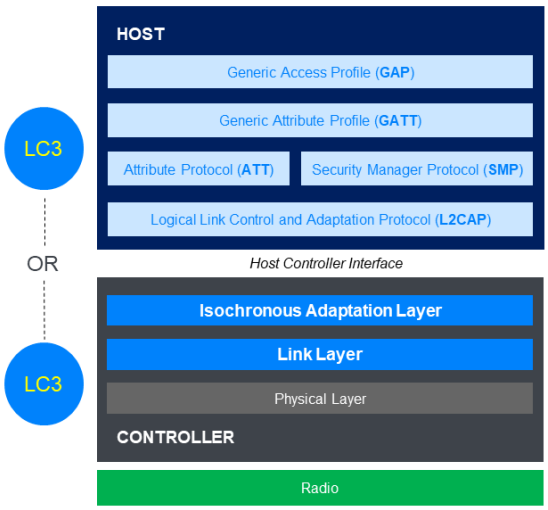


Figure 2 - The Bluetooth LE stack

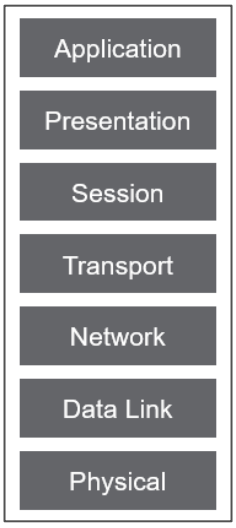


Figure 3 - The OSI Reference Model

Bluetooth mesh uses the Bluetooth LE controller with link layer and physical layer whilst the host part contains a collection of alternative layers which implement the Bluetooth mesh protocols and procedures. This resource does not cover Bluetooth mesh any further and those looking for an educational resource on this subject should consult section *16. Additional Resources below*.

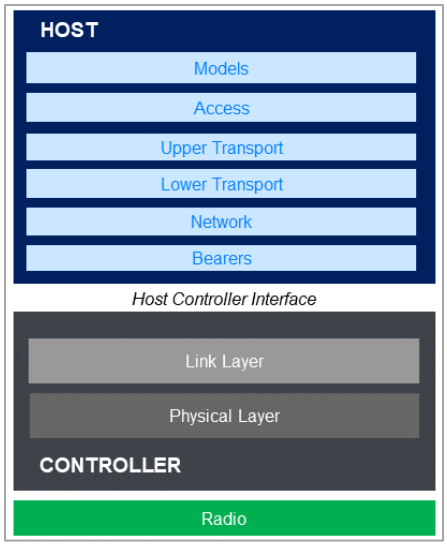


Figure 4 - The Bluetooth mesh stack

5.2 The Layers at a Glance

A summary of the key responsibilities and features of each layer of the Bluetooth LE stack as depicted in Figure 2 is as follows:

Layer	Key Responsibilities
Physical Layer	Defines all aspects of Bluetooth technology that are related to the use of radio (RF) including modulation schemes, frequency bands, channel use, transmitter and receiver characteristics. Several distinct, supported combinations of physical layer parameters are defined and are referred to as PHYs.
Link Layer	Defines air interface packet formats, bit stream processing procedures such as error checking, a state machine and protocols for over-the-air communication and link control. Defines several distinct ways of using the underlying radio for connectionless, connection-oriented and isochronous communication known as *logical transports.*
Isochronous Adaptation Layer **(ISOAL)**	Allows different frame durations to be used by devices using isochronous communication. Performs segmentation and reassembly of framed PDUs or fragmentation and recombination of unframed PDUs.
Host Controller Interface **(HCI)**	Provides a well-defined functional interface for bi-directional communication of commands and data between the host component and the controller. Supported by any one of several physical transport implementations.
Logical Link Control and Adaptation Protocol **(L2CAP)**	Acts as a protocol multiplexer within the host, ensuring protocols are serviced by the appropriate host component. Performs segmentation and reassembly of PDUs/SDUs between the layer below and the layer above L2CAP.
Security Manager Protocol **(SMP)**	A protocol used during the execution of security procedures such as pairing.
Attribute Protocol **(ATT)**	A protocol used by an ATT client and an ATT server which allows the discovery and use of data in the server's attribute table.
Generic Attribute Profile **(GATT)**	Defines high level data types known as services, characteristics and descriptors in terms of underlying attributes in the attribute table. Defines higher level procedures for using ATT to work with the attribute table.
Generic Access Profile **(GAP)**	Defines operational modes and procedures which may be used when in a non-connected state such as how to use advertising for connectionless communication and how to perform device discovery. Defines security levels and modes. Defines some user interface standards.

Each layer in the Bluetooth LE stack, as depicted in Figure 2 will now be examined in more detail.

6. The Physical Layer

The physical layer of Bluetooth LE defines how the radio transmitter/receiver is used to encode and decode digital data for transmission and receipt, and other radio-related parameters and properties which apply.

6.1 Frequency Band

Bluetooth LE operates in the 2.4GHz unlicensed band in the range 2400 MHz to 2483.5 MHz which is divided into 40 channels, each with a spacing of 2 MHz. How channels are used is defined by the Link Layer and the data transport architecture.

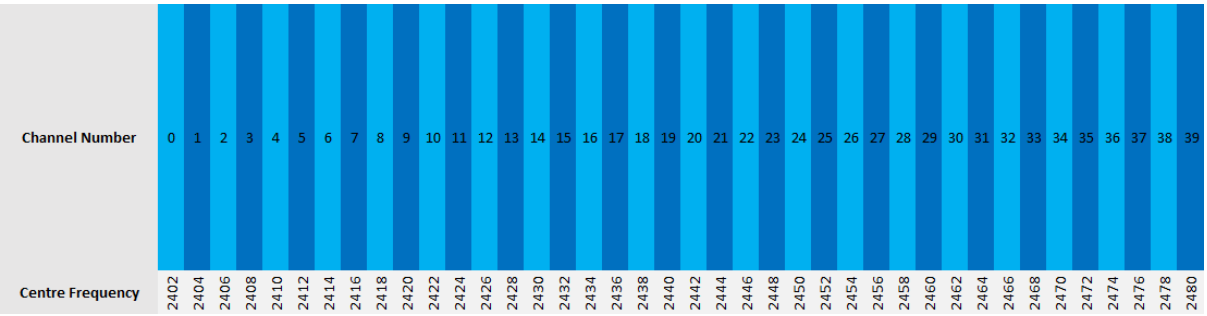


Figure 5 - Bluetooth LE channels

6.2 Modulation Scheme

To encode digital data from higher layers of the stack before transmission and to decode radio signals received, Bluetooth LE uses a modulation scheme called Gaussian Frequency Shift Keying (GFSK). GFSK works by taking a signal with the central frequency of the selected channel (the *carrier*) and shifting it up by a specified amount to represent a digital value of 1 or down by the same amount to represent a binary value of 0. Gaussian filtering is applied to the signal to reduce noise which may accompany abrupt frequency changes.

Figure 6 illustrates the basic frequency shift keying process. Note that the amount by which the frequency is shifted is known as the *frequency deviation* and is at least +/-185 kHz or at least 370 kHz depending on the PHY variant in use (*PHY* is explained in section 6.3).

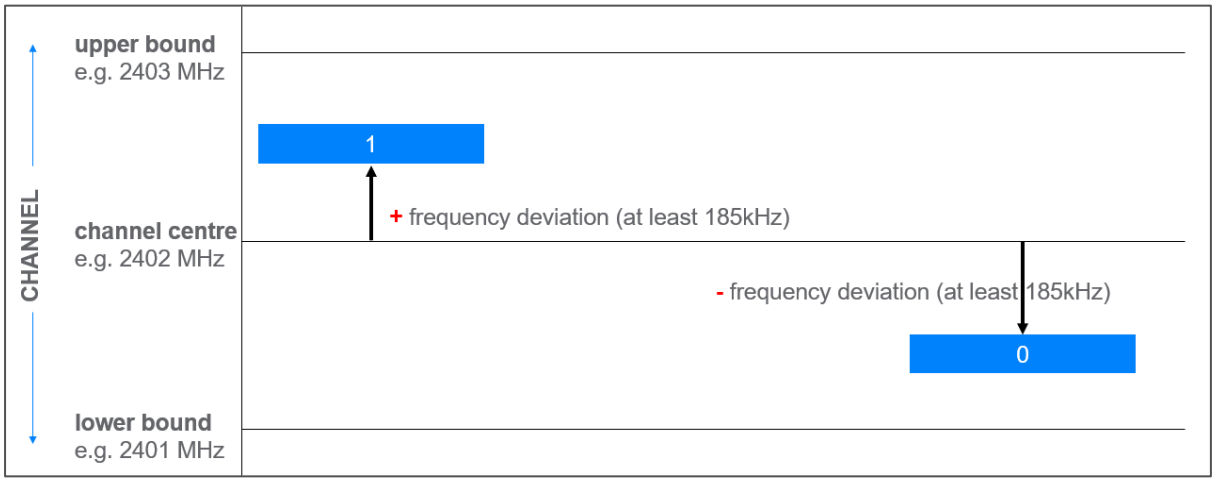


Figure 6 - Frequency Shift Keying in Bluetooth LE

6.3 PHY variants

Three modulation scheme variants are defined. Each variant is referred to as a *PHY* and has a name. Transmission speeds at the physical layer are measured in *symbols per second* rather than bits per second because the physical layer deals with analogue radio artefacts only, not digital concepts. Bluetooth LE uses a binary modulation scheme however meaning that a single analogue symbol represents a single digital bit higher in the stack.

The three PHY types defined are summarised as follows:

- The **LE 1M** PHY uses a symbol rate of 1 Msym/s with a required frequency deviation of at least 185 kHz and uses no special *coding*. All devices must support the LE 1M PHY.
- The **LE 2M** PHY is similar to LE 1M but uses a symbol rate of 2 Msym/s and has a required frequency deviation of at least 370 kHz. Support for the LE 2M PHY is optional.
- The **LE Coded** PHY uses a symbol rate of 1 Msym/s but packets are subject to a coding called Forward Error Correction (FEC) which is defined in the Link Layer. FEC increases the effective range of transmissions but reduces the application data rate. Support for the LE Coded PHY is optional.

A comparison of the three PHYs follows:

	LE 1M	**LE Coded S=2**	**LE Coded S=8**	**LE 2M**
Symbol Rate	1 Ms/s	1 Ms/s	1 Ms/s	2 Ms/s
Protocol Data Rate	1 Mbit/s	500 Kbit/s	125 Kbit/s	2 Mbit/s
Approximate Max. Application Data Rate	800 kbps	400 kbps	100 kbps	1400 kbps
Error Detection	CRC	CRC	CRC	CRC
Error Correction	NONE	FEC	FEC	NONE
Range Multiplier (approx.)	1	2	4	0.8
Requirement	Mandatory	Optional	Optional	Optional

Definitions

Term	**Definition**
Symbol Rate	The rate at which analogue symbols are transmitted at the physical layer.
Protocol Data Rate	The transmission rate of bits relating to Bluetooth protocol data units (PDUs) including their application data payload but excluding FEC data which is included in packets when the LE Coded PHY is in use.
Approximate Max. Application Data Rate	An approximate maximum rate at which application data can be communicated between applications on connected devices. Application data is transported in the payload part

	of various PDUs with the remainder of the protocol data rate being consumed by Bluetooth protocol data.
CRC	Cyclic Redundancy Check. A field used in the detection of transmission errors. This field and its use is defined in the Link Layer.

6.4 Time-Division

A Bluetooth LE radio is a half-duplex device, capable of transmitting and/or receiving but not both at the same time. However all PHYs are used in a Time Division Duplex (TDD) scheme so that the appearance of a full-duplex radio is given.

6.5 Transmission Power and Receiver Sensitivity

The Physical Layer defines transmitter characteristics including output power requirements for which the specification states that:

- the output power level at the maximum power setting shall be between 0.01 mW (-20 dBm) and 100 mW (+20 dBm).

Regulatory bodies in different parts of the world may override these requirements and implementers must ensure that devices are compliant with applicable local regulations.

Receiver sensitivity is defined as the receiver input level for which a particular Bit Error Rate (BER) is experienced. The BER specified varies according to the length of a received packet because the Link Layer appends a single Cyclic Redundancy Check (CRC) field to each packet and uses this as a mechanism for detecting one or more bits in error in the decoded packet. Since packets vary in length, and there is one CRC per packet, the length of the packet affects the calculated BER.

Typically quoted in discussions about Bluetooth LE receiver sensitivity is the BER of 0.1% which is the maximum error rate for packets up to 37 octets in length.

Other receiver characteristics defined by the Physical Layer include interference performance, out-of-band blocking, intermodulation characteristics, maximum usable input level and the required accuracy of received signal strength indications (RSSI) if supported.

6.6 Antenna Switching

Bluetooth LE supports two methods of calculating the direction from which a received signal was transmitted. The first is called Angle of Arrival (AoA) and the second, Angle of Departure (AoD). Both methods involve one device having an array of antennae and a process of switching from one antenna to another during the transmission of direction finding signals (AoD method) or when receiving signals (AoA method). Direction finding signals are standard Bluetooth packets which include a Constant Tone Extension (CTE) field.

Antenna arrays come in many different designs and switching from one antenna to the next can follow a range of different switching patterns. This is controlled by the host but the physical layer also defines some generally applicable rules about the process of antenna switching, related receiver requirements and some useful definitions.

The Bluetooth Core Specification covers this topic in more detail in Volume 6, Part A, section 5. More information on the AoA and AoD direction finding feature can be found in the Bluetooth Core Specification version 5.1 Feature Overview paper.

7. The Link Layer

7.1 Overview of the Link Layer

The Link Layer specification is almost the largest of the Bluetooth LE sections of the Bluetooth Core Specification, second only to the Host Controller Interface Functional Specification section. Arguably though, it is the most complicated.

The Link Layer has many responsibilities. It defines several types of packet that are transmitted over the air and an associated air interface protocol. Its operation is subject to a well-defined state machine. Depending on the state, the link layer may operate in a number of quite different ways, driven by events of a number of types. Numerous control procedures which affect the state of a link or link usage parameters are defined. Radio channel selection and classification are defined in the link layer specification.

The link layer supports both connected and connectionless communication and deterministic and (slightly) randomized event timing. It supports both point-to-point communication between two devices and one-to-many communication from one device concurrently to an unlimited number of receiving devices.

Much of the versatility of Bluetooth LE is rooted in the sophistication of the Link Layer.

7.2 Packets

The Link Layer defines two packet types. The first is used by the uncoded PHYs (see Figure 7), LE 1M and LE 2M and the second by the LE Coded PHY (see Figure 8).

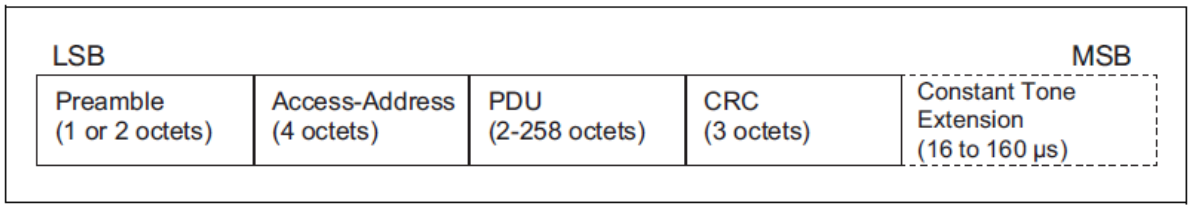


Figure 7 - Link layer packet format for the LE uncoded PHYs

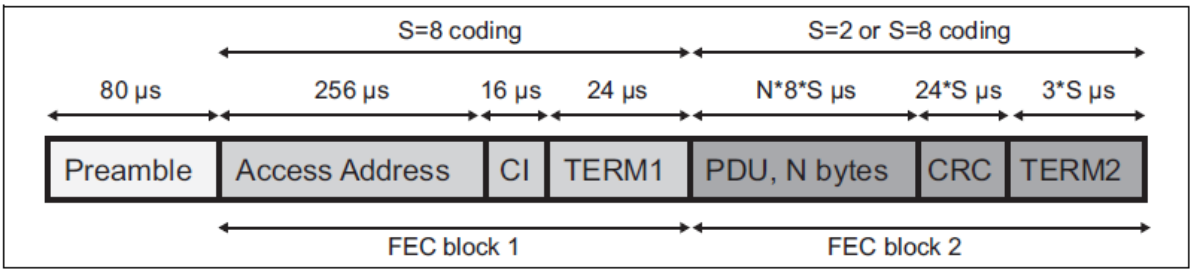


Figure 8 - Link layer packet for the LE Coded PHY

Both packet types include the fields *Preamble*, *Access Address*, and *CRC*. Table 1 explains each of these common fields.

Link Layer Packet Field Name	Description
Preamble	The preamble allows the receiver to synchronize precisely on the frequency of the signal, perform automatic gain control and estimate the symbol timing.

Access Address	The access address is used by receivers to differentiate signals from background noise and to determine the relevance or otherwise of a packet to the receiving device. For example, a pair of connected devices exchange packets with the same randomly allocated access address. Devices not participating in the connection will ignore such packets since the access address is not one that is relevant to them. Similarly, all legacy advertising packets use the same access address with a value of 0x8E89BED6 which indicates that these packets may be received by all devices.
CRC	The Cyclic Redundancy Check is used for error detection. Its value is calculated by the transmitter using the value of the other bits in the packet. On receiving a packet, the receiving device also calculates a CRC value from the values of bits in the received packet apart from those making up the CRC field. The receiver's calculated CRC is then compared with the value of the CRC field in the packet. If the two CRC values match then the packet was received correctly. If not, it is deemed to contain one or more bits in error.

Table 1 - Common Link Layer Packet Fields

The PDU field of Link Layer packets may contain a variety of different Protocol Data Units (PDUs) depending on how Bluetooth LE is being used. The Constant Tone Extension (CTE) is only present when one of the two direction finding methods (Angle of Arrival or Angle of Departure) is in use.

The PDU and CRC fields are subjected to a process called *whitening* before the packet is transmitted. The purpose of whitening is to avoid long sequences of zeroes or ones in packets as this can cause the receiver's frequency lock to drift. The whitening process is reversed by the receiver to restore the original bit stream before the CRC is checked.

The PDU field may be encrypted in which case it includes a Message Integrity Check field which protects against the PDU being tampered with[3].

When the LE Coded PHY is in use, the bit stream is subject to additional processing before transmission. The application of a Forward Error Correction (FEC) encoder followed by a Pattern Mapper generates additional data which is used by the receiver when applying these processes in reverse and where possible, correcting the value of any bits that have the incorrect value.

7.3 State Machine

The Link Layer is governed by a state machine which is shown in Figure 9.

[3] Section 15 covers security in Bluetooth LE in more detail

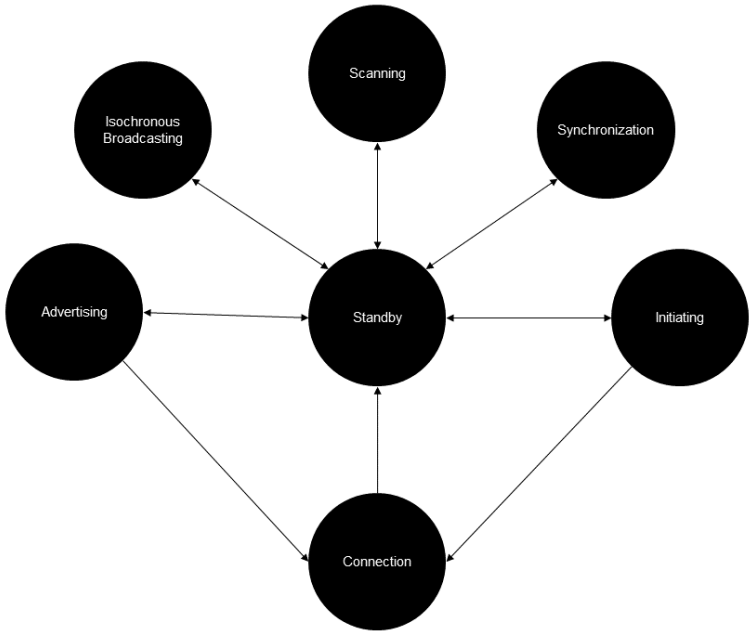


Figure 9 - The Link Layer State Machine

Refer to the Link Layer specification for full details of each state. A summary is presented in Table 2. Note that some terminology will be explained later in this section.

State	Description
Standby	Device neither transmits or receives packets.
Initiating	Responds to advertising packets from a particular device to request a connection.
Advertising	Transmits advertising packets and potentially processes packets sent in response to advertising packets by other devices.
Connection	In a connection with another device.
Scanning	Listening for advertising packets from other devices.
Isochronous Broadcast	Broadcasts *isochronous* data packets.
Synchronization	Listens for periodic advertising belonging to a specific *advertising train* transmitted by a particular device.

Table 2 - Link Layer states

When in the Connection state, two important device roles are defined. These are the Central role and the Peripheral role. A device which initiates a connection and transitions from the Initiating state to the Connection state assumes the Central role. A device which accepts a connection request, transitioning from the Advertising state to the Connection state assumes the Peripheral role.

Consider for example, a smartphone which includes a music player application and a portable Bluetooth LE speaker. Typically, the smartphone would assume the Central role and the speaker the Peripheral role. The smartphone discovers the speaker by scanning for its advertising packets and then, usually with the involvement of the user, initiates a connection to the speaker. Once connected, following additional procedures defined in the relevant LE Audio specifications, an audio stream is then established.

An instance of the state machine may only be in one state at a time. A link layer implementation may support more than one state machine instance concurrently.

Not all role and state combinations are permitted. The Bluetooth Core Specification has more details on this.

7.4 Channel Selection

As described in section 6.1 Frequency Band, Bluetooth LE divides the 2.4 GHz frequency band into 40 channels. The Link Layer controls how those channels are used and this in turn depends on the overall way in which Bluetooth LE is being used for communication (more formally, the current *physical channel* - this is covered in section 7.5 The Data Transport Architecture).

Bluetooth LE uses spread spectrum techniques in various different ways to communicate data via multiple channels over the course of time. This reduces the chances of collisions, making communication more reliable.

One well known example of a spread spectrum technique used in Bluetooth LE is that of *adaptive frequency hopping.* This involves the radio channel used for packet communication changing at regular intervals. Channels are chosen using a channel selection algorithm and a table of data called the *channel map* which classifies each channel as either *used* or *unused*. Implementations can monitor the quality of communication on each channel and if a channel is found to be performing badly, perhaps due to interference from other sources, the channel map can be updated to set that channel's classification to *unused* and this ensures that this channel is no longer selected by the algorithm. In this way, channel selection algorithm adapts to the conditions being experienced and optimizes for the most reliable performance.

How radio channels are used will be described further when discussing Bluetooth LE logical transports and their associated physical channels below.

7.5 The Data Transport Architecture

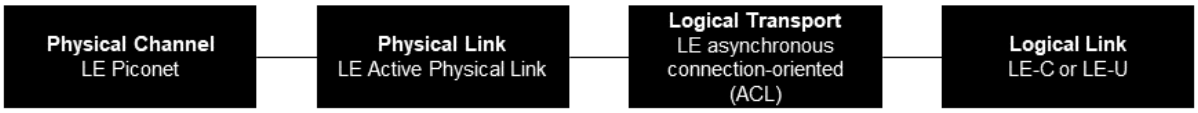
The architecture section of the Bluetooth Core Specification defines a number of concepts which collectively constitute the *Bluetooth data transport architecture*. Key amongst these concepts are the Physical Channel, Physical Link, Logical Link and Logical Transport. Certain combinations are defined for use in support of different application types, each with particular requirements regarding issues such as topology, timing, reliability and channel use.

A Physical Channel defines one of several different ways of communicating using Bluetooth. For example, communication can take place between two connected devices using the LE Piconet Physical Channel, which involves adaptive frequency hopping across 37 channels. Alternatively, the LE Advertising Physical Channel can be used for broadcast, connectionless communication from one device to an unlimited number of other devices. The LE Periodic Physical Channel can be used to broadcast data too, but on a regular basis with a deterministic time schedule. Observer (receiver) devices are able to determine that time schedule and use it to synchronize their scanning schedules.

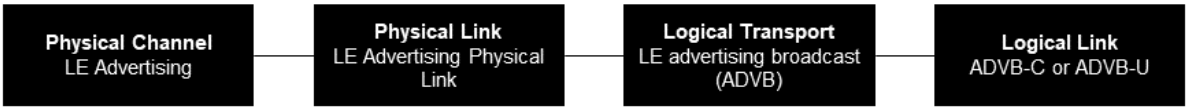
A Physical Link is based on a single, specific physical channel and specifies certain characteristics of that link such as the use or otherwise of power control.

Logical links and transports have various parameters which are designed to provide a suitable means of supporting a particular set of data communication requirements over a physical link, using a particular physical channel type.

For example, reliable, bi-directional, point to point communication in Bluetooth LE uses the LE asynchronous connection-oriented logical (ACL) transport with either a LE-C link for control data or a LE-U link for user data, over a physical link based on the LE Piconet Physical Channel.



On the other hand, unreliable, unidirectional broadcast communication in Bluetooth LE uses the LE Advertising Broadcast (ADVB) logical transport with either an ADVB-C link for control data or an ADVB-U link for user data, over a physical link based on the LE Advertising Physical Channel.



7.7 The Logical Transports

7.7.1 LE ACL - LE Asynchronous Connection-Oriented Logical Transport

7.7.1.1 Basics

When two Bluetooth LE devices are connected they are using the asynchronous connection-oriented logical transport (LE-ACL or simply ACL). LE-ACL is one of the most commonly used Bluetooth LE logical transport types, providing for connection-oriented communication of data. In fact, ACL connections are generally referred to simply as connections.

A device may establish a connection with an advertising device by responding to a received advertising packet with a PDU that requests a connection. A number of parameters are specified in the request. Amongst these parameters are *access address*, *connection interval*, *peripheral latency*, *supervision timeout* and *channel map*.

The device requesting the connection transitions from the Standby state to the Initiating state and then enters the Connection state and assumes the Central role. The other device transitions from Advertising to Connection and assumes the role of Peripheral.

The connection interval parameter defines how often in milliseconds, the radio can be used for servicing this connection. Whenever the connection interval expires, a *connection event* is said to begin and at this point, the Central device in the connection may transmit a packet. Connection events for a given connection each have a 16-bit identifier which is a counter value, incremented at each event. At the start of each connection event, the radio channel to be used is selected using the applicable channel selection algorithm.

The supervision timeout parameter specifies the maximum time which may elapse between two Link Layer data packets having been received before the link is considered to have been lost.

The Peripheral device, possessing the same agreed connection parameters as the Central device knows when to expect transmitted packets from the Central device and over which channel and so may choose to listen on that channel at precisely the same time and therefore receive the packet from the Central. The Peripheral may reply to the Central device 150 microseconds (+/- 2µs) after receiving the last bit of the Central's packet. Central and Peripheral then take turns, alternating between transmitting and receiving packets and may exchange an implementation-defined number

of packets during the connection event. Note that the Peripheral's behavior may be modified by a non-zero Peripheral Latency parameter value.

Figure 10 shows a basic exchange of packets, during two connection events with C>P indicating packet transmission by the Central device and P>C by the Peripheral.

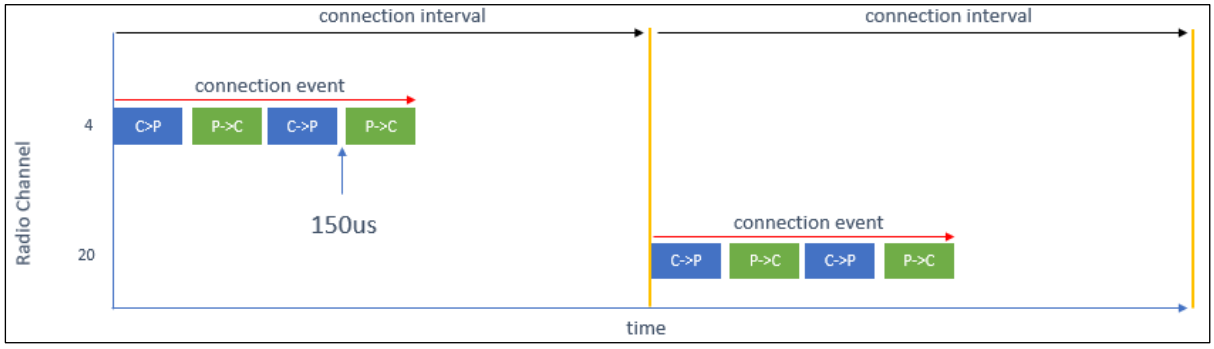


Figure 10 - Basic packet exchange over an LE-ACL connection

Packets exchanged over an LE ACL connection contain either LL Data PDUs or LL Control PDUs which are associated with Link Layer control procedures.

7.7.1.2 Ordering and Acknowledgements

LE-ACL incorporates a system which ensures that data is processed in the right order, that the receipt of packets can be acknowledged, and for this to be used to decide whether to move on to the next packet or instead, to retransmit the previous one.

Data packets contain three important fields which contribute to communication being reliable. These fields are called the Sequence Number (SN), Next Expected Sequence Number (NESN), and the More Data field. All three of these fields are single bit fields and their use provides a system of acknowledgements and a method for checking for the correct ordering of received packets.

Communication starts with the primary device (Device A) sending a link layer data packet with SN and NESN both set to zero. From this point on, at each packet exchange that takes place, if all is well, the value of the SN field as set by Device A, will alternate between zero and one. The general-purpose device (Device B) always knows therefore, what the SN value of the next packet to be received should be and checks for this.

If Device B receives a packet from Device A with the expected SN value, it responds with a link layer data packet that has NESN set to the logical value NOT(SN). So for example, if the received SN value was 1 then NESN in the response will be 0.

When Device A receives a response from Device B with NESN set to the value that Device A intends to use for SN in its next packet, Device A takes this to be an acknowledgement from Device B, confirming that it received the last transmitted packet correctly. Figure 11 shows this.

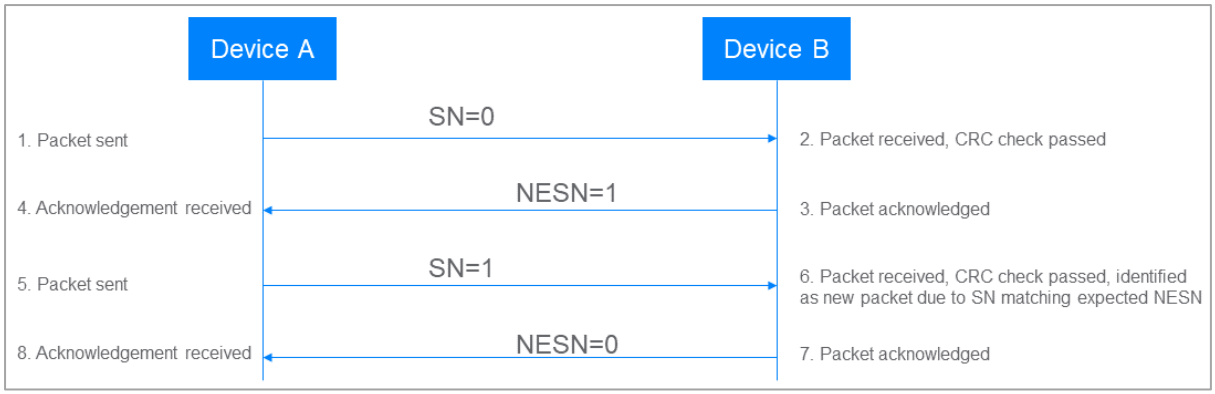


Figure 11 - A successful exchange of packets at the link layer

If Device B receives a packet with the wrong SN value, it assumes that the packet is the retransmission of the previous packet received, acknowledges it but does not pass it up the stack for further processing.

If Device A receives an unexpected NESN value in a reply from Device B or does not receive a reply at all, it resends the packet with the same SN value used originally. Different controller implementations are free to implement varying algorithms regarding how many times to resend before concluding communication to have failed. See Figure 12.

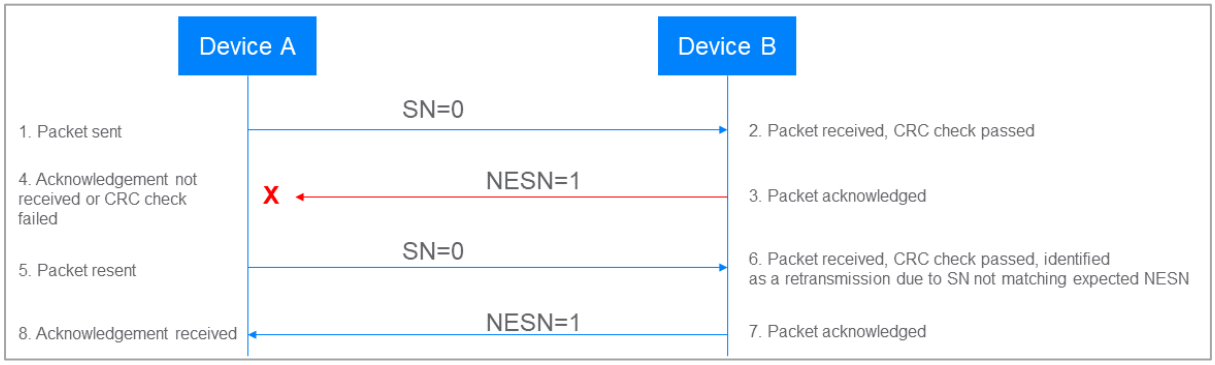


Figure 12 - Link Layer retransmissions

Each packet contains a CRC field and encrypted packets also contain an MIC field. On receiving a packet, the link layer checks the CRC and if present, the MIC. If either check fails, the packet is not acknowledged and this generally results in the originator of the packet resending it. See Figure 13.

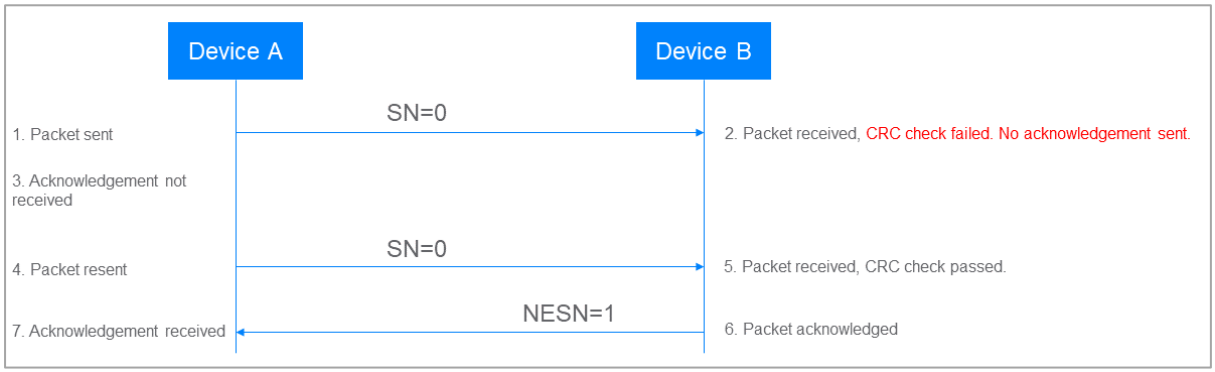


Figure 13 - Link Layer handling CRC failure

7.7.1.3 Peripheral Latency

The Peripheral is not required to listen for packets from the Central device during every connection event. The peripheral latency parameter defines the number of consecutive connection events during which the Peripheral does not have to be listening. This allows the Peripheral to save power.

Figure 14 shows the behaviour of the Peripheral with peripheral latency = 1 and therefore listening during alternate connection events only. The Central may transmit during those events where the Peripheral is not listening but such packets will not be received and therefore will not be acknowledged, ending the connection event.

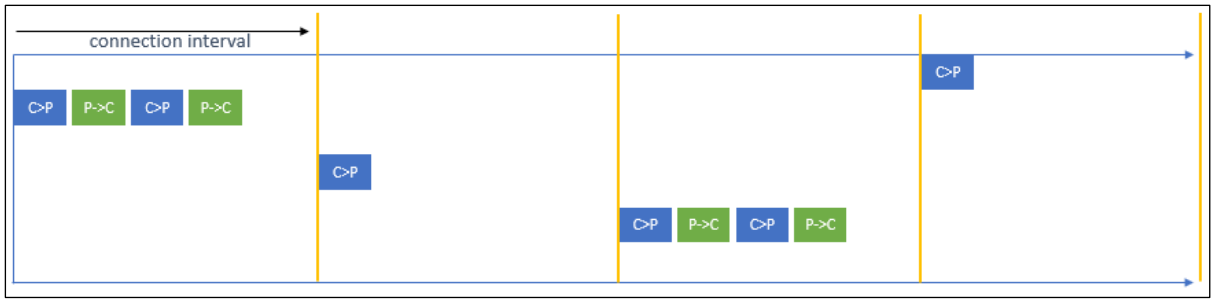


Figure 14 - An ACL connection with Peripheral Latency = 1

7.7.1.4 Channel Use

LE-ACL employs a scheme known as *adaptive frequency hopping*. At the start of each connection event, *frequency hopping* occurs, with a radio channel being deterministically selected from the set of available channels using a *channel selection algorithm*. Each device in the connection will then switch to the selected channel and over time and a series of connection events, communication will take place using a frequently changing series of different channels, distributed across the 2.4 GHz band, thereby significantly reducing the probability of collisions occurring.

Of the 40 channels defined for use by Bluetooth LE, 37 of these channels (known as the *general purpose channels)* are available for use by an LE-ACL connection.

In a given environment, some Bluetooth radio channels might not be functioning well, perhaps because interference is impacting them, whereas other channels are working reliably. Over time, the list of reliable channels and unreliable channels may change, as other wireless communication devices in the environment come and go.

The Central device in a connection maintains a *channel map* which classifies the general purpose channels as *used* or otherwise as *unused*. This channel map is shared with the Peripheral using a link layer procedure so that they each have the same information about which channels will be used and which will not. The channel selection algorithm ensures that channels designated as *unused* are avoided.

By default, all general purpose channels are designated *used* but Central devices may use implementation-specific techniques to monitor how well each channel is functioning. If the Central device determines that one or more channels are not working well enough, it can update their classification in the channel map to *unused*. Conversely, if a previously bad channel is found to be working well now, its classification can be updated in the channel map to *used*. Channel map updates may then be shared with the Peripheral device.

A Peripheral may also perform its own channel monitoring and at intervals, send *channel status reports* to the Central device, with each channel's status classified as *good*, *bad* or *unknown*. The

Central may then make decisions about channel classification in the channel map that take into account both its own radio conditions and those being experienced by the remote Peripheral device.

In this way, it is possible for a Bluetooth LE device to use only the optimal subset of available channels and so for example, coexist effectively with other wireless technologies that use statically allocated channels. This is the *adaptive* aspect of the Bluetooth *adaptive frequency hopping* system.

> ***Note**: Regulatory bodies may define adaptive frequency hopping and related terminology differently to the Bluetooth Core Specification. It is recommended that regulations regarding spectrum use in target markets are reviewed early in the product development lifecycle as this may inform certain implementation decisions.*

Figure 15 shows the way in which channels were used by two connected devices during testing and illustrates the way in which radio use is spread across the ISM 2.4 GHz spectrum. At the bottom of the chart you can see the *channel index* and frequencies in MHz. The *channel index* is an indirect way of referencing a radio channel.

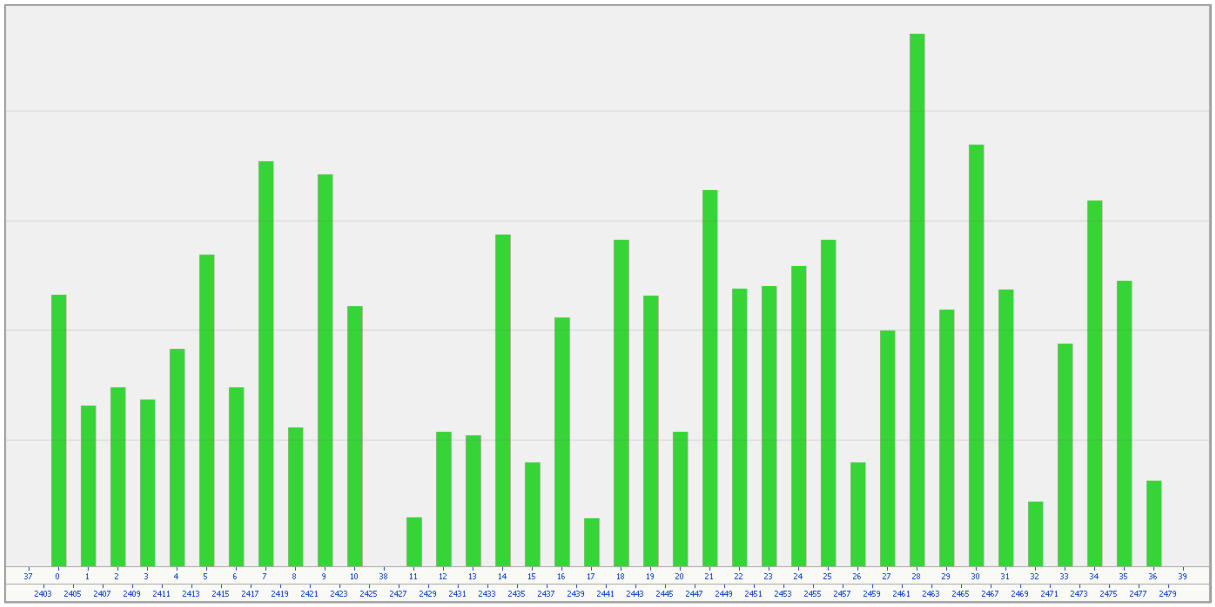


Figure 15 - Adaptive Frequency Hopping distributing communication across channels

7.7.1.5 Link Layer Control

The Link Layer specification specifies a number of control procedures. A selection of examples appear in Table 3.

Control Procedure	**Description**
Connection Update	Allows either the Central or Peripheral device to request changes to the connection parameters *connection interval*, *peripheral latency* and *supervision timeout*.
Channel Map Update	Allows the Central device to transfer its latest channel map data to the connected Peripheral.
Encryption	Allows either Central or Peripheral to enable the encryption of packets.
Feature Exchange	Allows Central or Peripheral to initiate an exchange of the Link Layer features each device supports, encoded as a bitmap field.

Periodic Advertising Sync Transfer	Allows either Central or Peripheral to transfer periodic advertising[4] synchronization information relating to a periodic advertising train that has been discovered to the other device over an LE ACL connection.
CIS Creation Procedure	Allows a Central device to create a Connected Isochronous Stream (CIS)[5] with the Peripheral.

[4] Periodic advertising is explained in 7.7.3 PADVB -

[5] Isochronous Streams are explained in 7.7.4 PAwR - LE Periodic Advertising with Responses

7.7.4.1 Basics

PAwR is similar to periodic advertising (PADVB) in several ways:

- PADVB allows application data to be transmitted by one device (the Broadcaster) to one or more receiving devices (the *Observers*), forming a one-to-many communication topology. The same is true of PAwR.
- PAwR and PADVB both use a connectionless communication method.
- Transmission of advertising packets is periodic with a fixed interval and no random perturbation of the schedule in both cases.
- Observers can establish the periodic transmission schedule used by the Broadcaster from AUX_ADV_IND PDUs or by using the Periodic Advertising Sync Transfer (PAST) procedure.

PAwR differs from PADVB as follows:

- PADVB supports the unidirectional communication of data from a Broadcaster to Observers only. PAwR Observers can transmit response packets back to the Broadcaster. PAwR provides a *bidirectional*, connectionless communication mechanism.
- Synchronization information for periodic advertising *without* responses (PADVB) is contained within the SyncInfo field of AUX_ADV_IND PDUs. Synchronization information for periodic advertising *with* responses (PAwR) is contained within the SyncInfo field and in the ACAD field of AUX_ADV_IND PDUs.
- The PADVB Broadcaster schedules transmissions within advertising *events*. The PAwR Broadcaster schedules transmissions in a series of events *and subevents,* and Observers are expected to have synchronized in such a way as to listen during a specific subevent or subevents only.
- The PAwR Broadcaster may use a transmission time slot to send a connection request (AUX_CONNECT_REQ PDU) to a specific device and establish an LE-ACL connection with it. PADVB does not have this capability.
- With periodic advertising without responses (PADVB), application data tends to only change from time to time. PAwR is designed with the expectation that application data will change frequently.
- With PADVB, the same application data is delivered to all Observer devices synchronized to the same advertising set. With PAwR, different data can be delivered to each Observer device or set of Observer devices.

Support for the Periodic Advertising Sync Transfer (PAST) procedure is optional with PADVB but mandatory with PAwR.

7.7.4.2 Channel Use

Channel selection is accomplished using Channel Selection Algorithm #2, and takes place at each periodic advertising subevent (see *7.7.4.3 Scheduling*). Responses to PDUs transmitted in a subevent use the same channel. This includes AUX_SYNC_SUBEVENT_RSP PDUs sent in response to a AUX_SYNC_SUBEVENT_IND PDU and AUX_CONNECT_RSP PDUs which are sent in response to AUX_CONNECT_REQ PDUs.

7.7.4.3 Scheduling

As with other advertising modes, activity takes place in events which in the case of PAwR are known as *Periodic advertising with responses events* (PAwR events). These events occur at fixed intervals, with no random perturbation in scheduling. An event starts every *periodic advertising interval* ms.

Each PAwR event consists of several subevents, and it is during subevents that advertising packets are transmitted. The Host configures the number of subevents per event up to a maximum of 128. A subevent starts every *periodic advertising subevent interval* ms. The Host configures the number of subevents per event and the periodic advertising subevent interval using a Host Controller Interface (HCI) command called HCI_LE_Set_Periodic_Advertising_Parameters V2 (or later).

See *Figure 23 - PAwR events and subevents*.

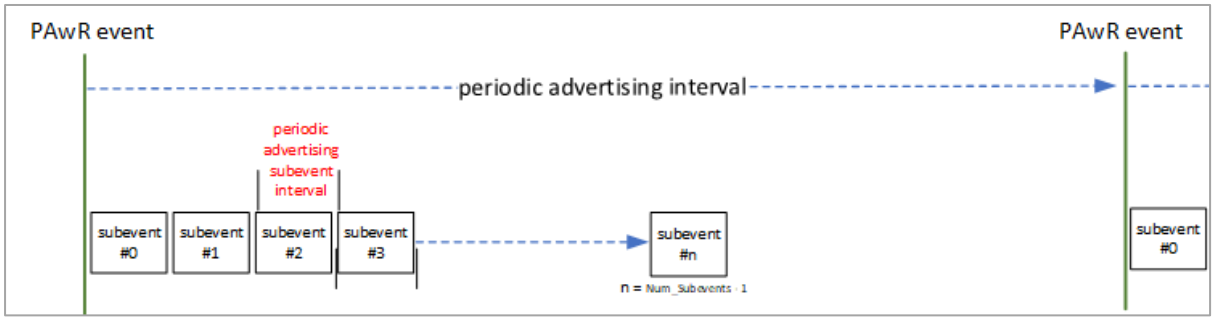


Figure 23 - PAwR events and subevents

In each subevent, the Broadcaster transmits one packet, which usually contains an AUX_SYNC_SUBEVENT_IND PDU but may instead contain an AUX_CONNECT_REQ PDU. After a delay, known as the *Periodic Advertising Response Slot Delay*, a series of time slots are reserved within the same subevent for receiving responses from Observer devices. Responses to AUX_SYNC_SUBEVENT_IND PDUs are sent in AUX_SYNC_SUBEVENT_RSP PDUs. The Host configures the number of response slots required by the HCI command HCI_LE_Set_Periodic_Advertising_Parameters. Figure 24 illustrates the structure of a PAwR subevent.

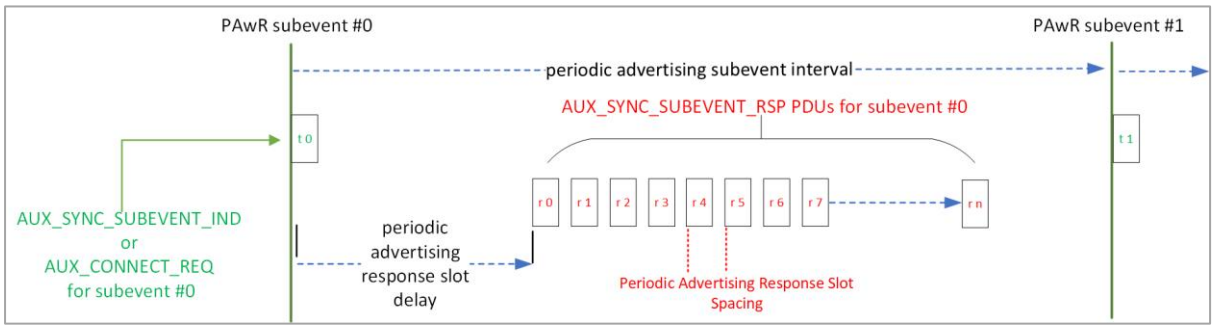


Figure 24 - A PAwR subevent with response slots

7.7.4.4 Synchronization Establishment

General

The process of *synchronizing* provides the Observer device with the information it needs to efficiently scan for and receive relevant packets transmitted by the advertising device. In the case of PAwR, there are three aspects to this:

1. The Observer needs to know how often *periodic advertising with responses event*s will occur and when the next such event will occur. This information is provided in a parameter called the *periodic advertising interval* and a calculated value known as *syncPacketWindowOffset*.

2. The Observer needs information about *subevents,* including how often they occur and how many subevents each *periodic advertising with responses event* accommodates. It also needs to know certain details relating to the time slots within each subevent reserved for the transmission of responses. This information is contained within parameters known as *Subevent_Interval, Num_Subevents*, *Response_Slot_Delay*, *Response_Slot_Spacing*, and *Num_Response_Slots*.

3. Finally, an Observer needs to know which subevent number it should scan for, which particular response slot it should use, and the access address to use in response packets transmitted.

Having acquired the event timing information described in (1) and the subevents information in (2), the Observer has a complete description of the timing parameters and structure of the events and subevents of the PAwR advertising train. But it is only when it has the information in (3) that it can schedule its scanning such that it receives only those packets that are expected to contain data of relevance and can schedule the transmission of response packets.

(1) and (2) are dealt with by the PAwR logical transport, as defined in the Bluetooth Core Specification. There is a choice of two procedures that may be used to obtain this level of synchronization information. The two procedures are covered in this paper in sections *Scanning for Periodic Advertising Synchronization Information* and *Periodic Advertising Sync Transfer (PAST)*.

(3) must be addressed by the application layer and may be defined in an applicable Bluetooth profile specification such as the Electronic Shelf Label (ESL) profile.

Power Control Request	Allows one peer to request that the other peer adjust its transmit power level.
Channel Classification Reporting	Allows a Peripheral to report channel classification data to the connected Central.

Table 3 - Example Link Layer control procedures

7.7.1.6 Subrated Connections

Subrated connections are LE ACL connections which have additional properties assigned to them and behave differently in some ways. The additional properties are called the *subrate factor*, *subrate base event*, and *continuation number*.

Scanning for Periodic Advertising Synchronization Information

PAwR and PADVB each use a similar procedure for acquiring periodic advertising synchronization information by scanning.

With both PAwR and PADVB, an Observer scans for AUX_ADV_IND packets transmitted on the secondary advertising channels. AUX_ADV_IND includes the SyncInfo field, which contains the *periodic advertising interval* value and some data items from which to calculate a variable called *syncPacketWindowOffset*. Having acquired these two values, the Observer can calculate when *periodic advertising with responses event*s will occur, per (1) in *General*.

PAwR also requires information about subevents and response slots, per (2) in *General,* before it can complete the synchronization procedure. This information is to be found in the same AUX_ADV_IND PDU from which the *periodic advertising interval* was obtained but in an advertising data type (AD type) called Periodic Advertising Response Timing Information which itself is to be found in the Additional Controller Advertising Information (ACAD) field of the PDU.

Periodic Advertising Sync Transfer (PAST)

When using the PAST procedure, sometimes the device passing the synchronization parameters over the connection will first acquire it by scanning on behalf of the other device. In the case of PAwR, however, support for PAST is mandatory and so the PAwR Broadcaster can pass the required synchronization data over an LE ACL connection to the Observer. If this approach is taken, no scanning for AUX_ADV_IND PDUs is necessary by either device.

Subevent Synchronization and Response Slot Allocation

Subevent synchronization is concerned with indicating to an Observer device the subevent it should perform scanning for. One or more Observer devices may be synchronized to the same subevent. An individual Observer may be synchronized to receive during one or more subevents.

In addition, for an Observer to be able to send a response PDU, it must have some basis for determining which subevent response slot to use.

Both of these concerns are the responsibility of the application layer.

The subrated connection properties provide a mechanism for indicating that only a specific subset of connection events are to be actively used by the connected devices, with the radio not being used in other connection events. A subrated connection can therefore have a short ACL connection interval but still exhibit a low duty cycle.

Figure 16 illustrates the basic concepts relating to subrated connections.

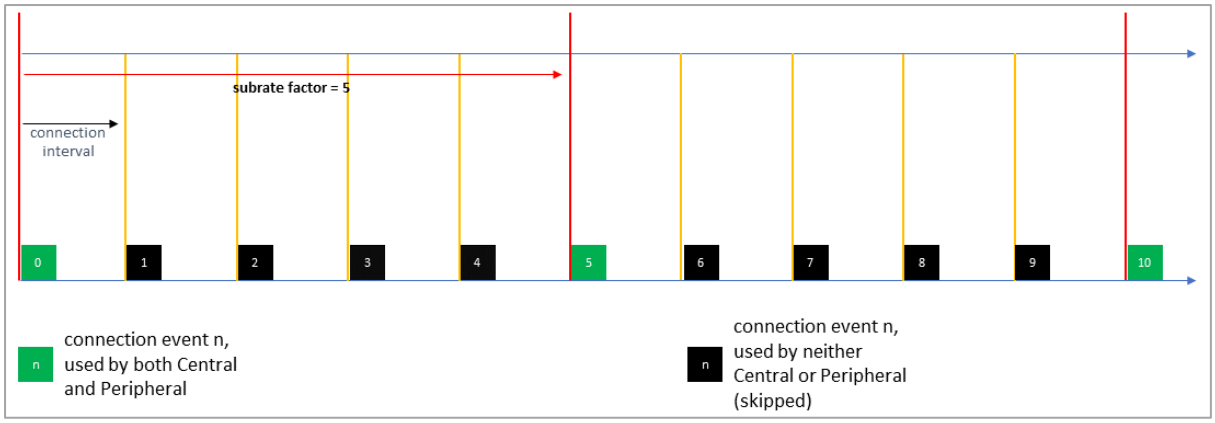


Figure 16 - A basic subrated connection with subrate factor=5

Here we can see that only one in every five connection events is used. The other four are skipped and so there is no radio activity during those connection events. This ratio of used to skipped connection events is determined by the *subrate factor* parameter and in this example it is set to 5.

The connection event during which the radio is used to transmit and receive link layer packets is known as a *subrated connection event*.

Given the relationship between the underlying ACL connection parameters and those that govern connection subrating, a subrated connection can be thought of as having both a connection interval which controls the frequency at which ACL connection events occur and an *effective connection interval*, which determines how often those ACL connection events are actually used, after the subrating parameters have been applied.

Subrated connections use a different set of Link Layer control procedures and in particular, the procedure for updating subrated connection parameters works differently to the general Control Update procedure. Critically, changes to subrated connection parameters can be applied almost instantaneously whereas general parameter changes can take a significant amount of time to take effect. The advantage of subrated connections therefore is that persistent connections which exhibit a low duty cycle and consume little power can be established and can be switched to a high duty cycle, high bandwidth connection with no delay that any user could notice. This capability has particular applicability in some LE Audio scenarios such as those involving hearing aids and smartphones.

The Bluetooth Core Specification Version 5.3 Feature Enhancements paper has a substantial chapter dedicated to the subject of subrated connections and is recommended as a source of further information.

7.7.2 ADVB - LE Advertising Broadcast

7.7.2.1 Basics

LE Advertising Broadcast (or simply *advertising*) provides a connectionless communication mode. It may be used to transfer data or to indicate the availability of a Peripheral device to be connected to.

Generally, advertising packets are intended to be received by any scanning device in range and as such, advertising may be used to concurrently transfer data to multiple scanning devices in a one-to-many topology. A special form of advertising known as *directed advertising* is defined however and this allows the connectionless communication of data from one advertising device to one specific scanning device, identified by its Bluetooth device address.

Advertising itself supports the communication of data in one direction only, from the advertising device to scanning devices but such devices may reply to advertising packets with PDUs that request further information or for a connection to be formed. When a scanning device replies to obtain further information it is said to be performing *active scanning*. When it does not, it is said to be performing *passive scanning*.

Advertising is generally referred to as an unreliable transport since no acknowledgements are sent by receivers.

Two categories of advertising procedure are defined and are referred to in the Bluetooth Core Specification as *legacy advertising* and *extended advertising*.

7.7.2.2 Legacy Advertising

7.7.2.2.1 Channel Use and Packet Size

Legacy advertising packets using the ADV_IND PDU type (see 7.7.2.2.3 Legacy Advertising and Associated PDU Types) are 37 octets long with a 6 octet header and a payload of at most 31 octets. Identical copies of advertising packets are transmitted on up to three dedicated channels numbered 37, 38 and 39 known as the *primary advertising channels*, one channel at a time and in some sequence.

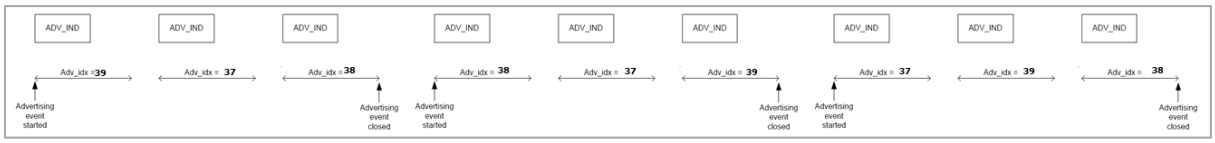


Figure 17 - legacy advertising and channel use

7.7.2.2.2 Scheduling

The transmission of an advertising packet takes place whenever an *advertising event* occurs. The scheduling of advertising events is controlled by timing parameters and in the basic case is deliberately made slightly irregular so as to avoid persistent collisions with other advertising devices. A value known as *advDelay* is assigned a pseudo-random value in the range 0 - 10ms at each advertising event and this is added to the regular *advertising interval* (*advInterval*) so that advertising events are perturbed in time. Figure 18 reproduces Figure 4.5 from Volume 6 Part B of the Bluetooth core specification and illustrates the effect of the advDelay parameter.

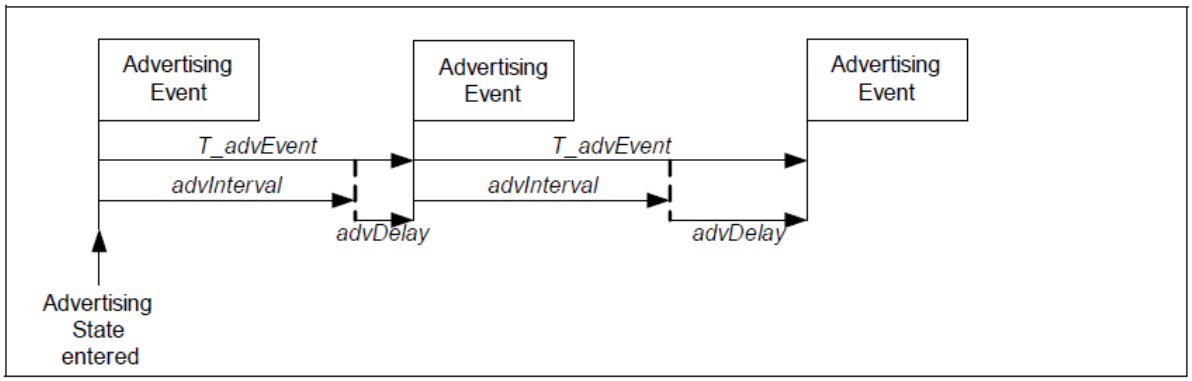


Figure 18 - Advertising events perturbed in time using advDelay

Scheduling advertising events in this way helps avoid collisions but makes it harder for receivers to efficiently receive advertising packets, requiring a higher RX duty cycle to accommodate the unpredictable timing of advertising events.

7.7.2.2.3 Legacy Advertising and Associated PDU Types

Several PDU types are defined for use with legacy advertising. Different types of PDU are used for undirected advertising, where packets are destined for any scanning device and for directed advertising, where packets are addressed to one specific device. The PDU type also indicates whether or not active scanning is allowed, with receivers replying with requests for further data and whether or not the advertising device may be connected to. All legacy advertising takes place on one or more of the primary channels numbered 37, 38 and 39 and may only use the LE 1M PHY.

Table 4 lists the legacy advertising PDUs.

PDU Name	Description	Channels	PHY(s)	Transmitted By	Scannable	Connectable
ADV_IND	Undirected advertising	primary	LE 1M	Peripheral	Y	Y
ADV_DIRECT_IND	Directed advertising	primary	LE 1M	Peripheral	N	Y
ADV_NONCONN_IND	Undirected, non-connectable, non-scannable advertising	primary	LE 1M	Peripheral	N	N
ADV_SCAN_IND	Undirected, scannable advertising	primary	LE 1M	Peripheral	Y	N
SCAN_REQ	Scan request	primary	LE 1M	Central	N/A	N/A
SCAN_RSP	Scan response	primary	LE 1M	Peripheral	N/A	N/A
CONNECT_IND	Connect request	primary	LE 1M	Central	N/A	N/A

Table 4 - Legacy Advertising PDUs

Section 4.4 of the Link Layer Specification chapter within the Bluetooth Core Specification gives full details of all advertising PDU types.

7.7.2.3 Extended Advertising

Bluetooth Core Specification version 5 introduced some major changes to how advertising can be performed. Eight new PDUs relating to advertising, scanning, and connecting were added and new procedures defined. Collectively this new set of advertising capabilities is known as *extended advertising*.

Extended advertising allows much larger amounts of data to be broadcast, advertising to be performed to a deterministic schedule and multiple distinct sets of advertising data governed by different configurations to be transmitted. It offers significant improvements regarding contention and duty cycle too.

Extended advertising is used by both the ADVB and the PADVB logical transports.

7.7.2.3.1 Channel Use and Packet Size

Radio channels are used differently to the way they are used when performing legacy advertising with primary advertising channels 37, 38 and 39 carrying less data and general-purpose channels 0 - 36 carrying most of the data.

As described in 7.7.2.2 Legacy Advertising, legacy advertising transmits the same payload up to three times on three different primary advertising channels. Extended advertising transmits payload data once only, with small headers referencing it from the primary channels. The total amount of data transmitted is therefore less than in the equivalent case using legacy advertising and so the effective duty cycle is reduced.

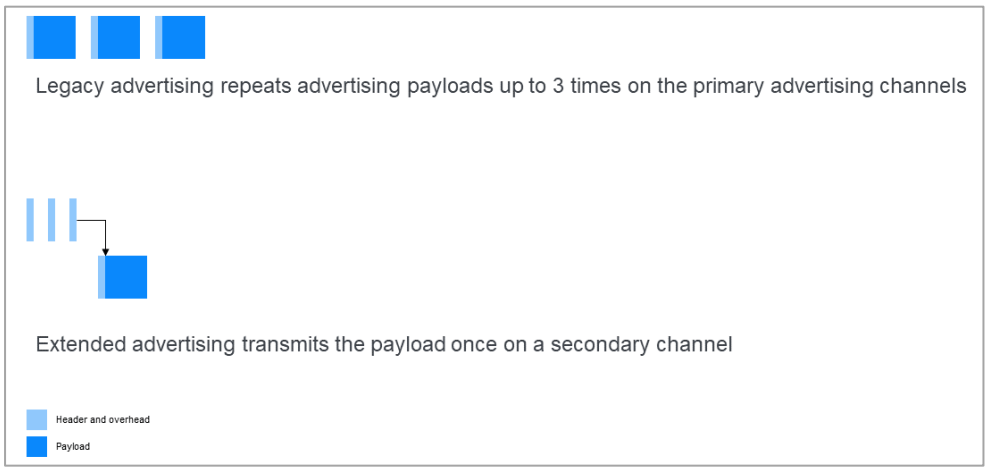


Figure 19 - Reduced contention and duty cycle

Extended advertising allows packets to be up to 255 octets long. This is accomplished in part through offloading the payload to one of the general-purpose channels in the 0-36 channel number range.

Figure 20 - Extended advertising supports larger advertising packets and channel offload

When performing extended advertising only header data is transmitted on the *primary channels* numbered 37, 38 and 39. This includes a field called AuxPtr.

The AuxPtr field references an associated *auxiliary packet* containing the payload which will be transmitted on a general-purpose channel from the set of channels numbered 0 - 36. AuxPtr includes the general-purpose channel index indicating the channel that the auxiliary packet will be transmitted on so that receivers know where to find it. Packets transmitted on a general-purpose channel and referenced by the AuxPtr field from a packet on the primary channels are known as *subordinate packets* whilst the referencing packet is known as a *superior packet*.

Selection of channel index values in AuxPtr is implementation-specific with the Bluetooth Core Specification only recommending that “sufficient channel diversity is used to avoid collisions”.

7.7.2.3.2 Packet Chaining

For those use cases where an application needs to broadcast even more data (up to 1,650 bytes), it is possible for the controller to fragment data and chain packets together with each packet

containing a subset of that data. Each chained packet can be transmitted on a different channel, with the AuxPtr header field referencing the next in the chain. Figure 21 illustrates this.

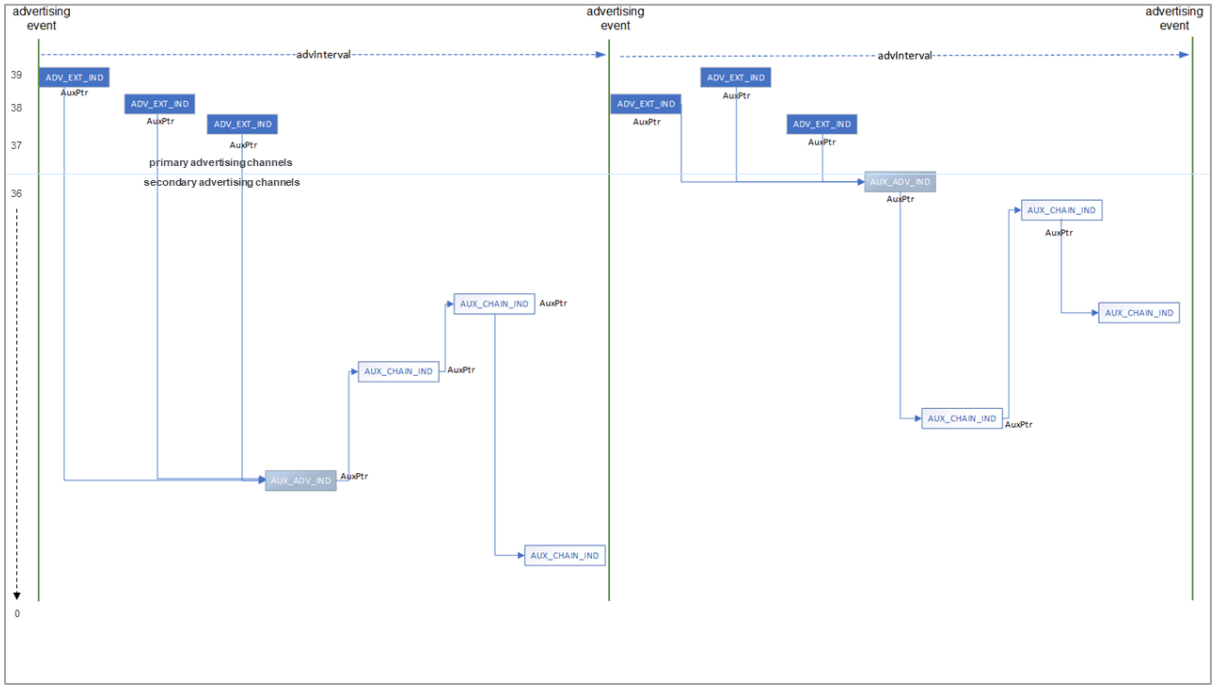


Figure 21 - Extended advertising with packet chaining

7.7.2.3.3 Advertising Sets

Legacy advertising does not make formal provision for advertising payload and parameters to vary. Extended advertising includes a standard mechanism for having multiple, distinct sets of advertising data.

Advertising sets have an ID which is used to indicate which set a given packet belongs to and each set has its own advertising parameters, such as its advertising interval and the PDU type to be used.

The task of scheduling and transmitting the different sets falls to the Link Layer in the Controller rather than it having to be driven by the Host, which would be far less power efficient. The Host needs only to inform the Controller of the advertising sets and their respective parameters initially, after which the Link Layer takes over.

7.7.2.3.4 Periodic Advertising

Extended advertising includes a method of advertising which uses deterministic scheduling, the details of which may be discovered and synchronized to by scanning devices. This is called *Periodic Advertising*. Periodic Advertising is defined as a distinct logical transport and so is described in section 7.7.3 PADVB - LE Periodic Advertising Broadcast.

7.7.2.3.5 Extended Advertising and Associated PDU Types

A number of PDU types are defined for use with extended advertising. Table 5 lists the extended advertising PDUs.

PDU Name	Description	Channels	PHY(s)	Transmitted By
ADV_EXT_IND	Extended advertising	primary	LE 1M, LE Coded	Peripheral
AUX_ADV_IND	Subordinate extended advertising	general-purpose	LE 1M, LE 2M, LE Coded	Peripheral
AUX_CHAIN_IND	Additional advertising data	general-purpose	LE 1M, LE 2M, LE Coded	Peripheral
AUX_SYNC_IND	Periodic advertising synchronization	periodic	LE 1M, LE 2M, LE Coded	Peripheral
AUX_SCAN_REQ	Auxiliary scan request	general-purpose	LE 1M, LE 2M, LE Coded	Central

AUX_SCAN_RSP	Auxiliary scan response	general-purpose	LE 1M, LE 2M, LE Coded	Peripheral
AUX_CONNECT_REQ	Auxiliary connect request	general-purpose	LE 1M, LE 2M, LE Coded	Central
AUX_CONNECT_RSP	Auxiliary connect response	general-purpose	LE 1M, LE 2M, LE Coded	Peripheral

Table 5 - Extended Advertising PDUs

The payload of PDUs of type ADV_EXT_IND, AUX_ADV_IND, AUX_SCAN_RSP, AUX_SYNC_IND, AUX_CHAIN_IND and AUX_CONNECT_RSP are defined by the same format known as the Common Extended Advertising Payload Format. This includes fields such as the AuxPtr field and AdvMode. AdvMode uses two bits to indicate the connectable and scannable properties of the advertising mode rather than distinct PDU types.

Section 4.4 of the Link Layer Specification chapter within the Bluetooth Core Specification gives full details of all advertising PDU types.

7.7.2.3.6 Scheduling

Extended advertising takes place in *extended advertising events*. An extended advertising event starts at the same time as an advertising event and includes the superior packet with an AuxPtr field and the subordinate packets related to it.

7.7.2.4 Comparing Legacy Advertising and Extended Advertising

Table 6 presents a summary comparison of legacy advertising and extended advertising.

	Legacy Advertising	**Extended Advertising**	
Max. host advertising data size	31 bytes	1,650 bytes	Extended Advertising supports *fragmentation* which enables a 50x larger maximum host advertising data size to be supported.
Max. host advertising data per packet	31 bytes	254 bytes	Extended Advertising PDUs use the *Common Extended Advertising Payload Format* which supports an 8x larger advertising data field.
TX channels	37,38,39	0-39	Extended Advertising uses the 37 general-purpose channels as secondary advertising channels. The ADV_EXT_IND PDU type may only be transmitted on the primary advertising channels (37, 38, 39) however .
PHY support	LE 1M	LE 1M LE 2M (excluding ADV_EXT_IND PDUs) LE Coded	All Extended Advertising PDUs except for ADV_EXT_IND may be transmitted using any of the three LE PHYs except for the ADV_EXT_IND PDU which may be transmitted using the LE 1M or LE Coded PHYs.
Max. active advertising configurations	1	16	Extended Advertising includes *Advertising Sets* which enable advertising devices to support up to 16

			different advertising configurations at a time and to interleave advertising for each advertising set according to time intervals defined in the sets.
Communication types	Asynchronous	Asynchronous Synchronous	Extended Advertising includes *Periodic Advertising*, enabling time-synchronised communication of advertising data between transmitters and receivers.

Table 6 - Comparing legacy and extended advertising

7.7.3 PADVB - LE Periodic Advertising Broadcast

7.7.3.1 Basics

Advertising as performed by the ADVB logical transport (see 7.7.2 ADVB - LE Advertising Broadcast) includes a degree of randomness in the timing of advertising packet transmission. Random delays of between 0 and 10ms are deliberately inserted in the scheduling of advertising events to help avoid persistent packet collisions. When performing legacy advertising this is the only way in which advertising can work.

Periodic advertising involves the transmission of packets to a deterministic schedule and provides a mechanism which allows other devices to synchronize their scanning for packets with the schedule of the advertising device. Periodic advertising is always non-scannable and non-connectable.

Periodic advertising can benefit observer devices by offering a more power-efficient way to perform scanning and is a key component in LE Audio broadcast solutions.

Advertising takes place at fixed intervals called the *periodic advertising interval* and the advertising data payload may change. A series of AUX_SYNC_IND and associated AUX_CHAIN_IND PDUs is said to form a *periodic advertising train*.

At each periodic advertising event a single AUX_SYNC_IND PDU is transmitted followed by zero or more AUX_CHAIN_IND PDUs depending on whether or not the host-provided payload requires fragmentation.

The AUX_ADV_IND PDU includes a field called SyncInfo which is part of the Common Extended Advertising Payload Format and which contains channel and timing offset information.

7.7.3.2 Channel Use

Periodic Advertising uses the 37 general-purpose advertising channels. A channel is selected at the start of each periodic advertising event using channel selection algorithm #2 with an event counter field called paEventCounter as input. This counter is incremented at each periodic advertising event. Any auxiliary AUX_CHAIN_IND PDUs related to a AUX_SYNC_IND PDU have their channel selected using an implementation-specific algorithm and specified in the AuxPtr field. See Figure 22.

7.7.3.3 Scheduling

The *periodic advertising interval* determines how often periodic advertising for a given advertising set can occur. It starts with the transmission of an AUX_SYNC_IND PDU and continues with a series of zero or more AUX_CHAIN_IND PDUs as shown in Figure 22.

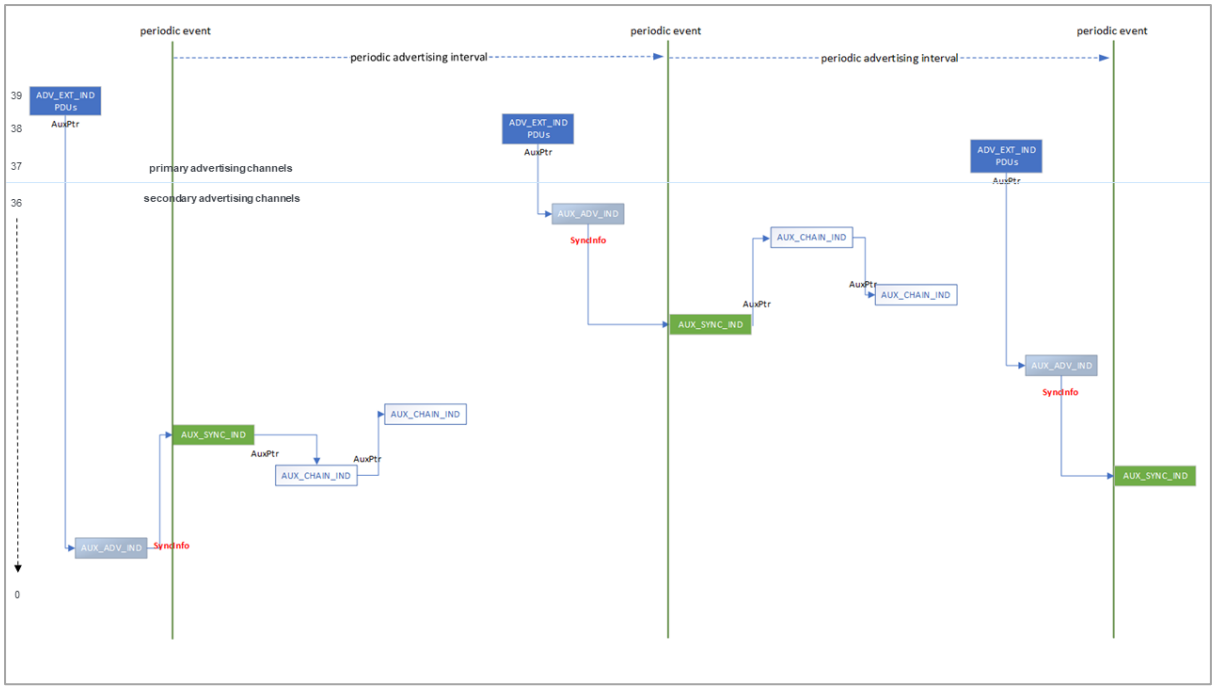


Figure 22 - Periodic advertising events

Note that Figure 22 has been simplified with potentially multiple ADV_EXT_IND PDUs on different primary advertising channels represented by a single box.

7.7.3.4 Synchronization Establishment

A scanning device may synchronize with a periodic advertising train in either of two ways. It may either scan for AUX_ADV_IND PDUs and use the contents of the SyncInfo field to establish the periodic advertising intervals, timing offset and channel information to be used or it may receive this information over an LE-ACL connection from another device which has itself determined this information from AUX_ADV_IND PDUs. This method is known as the Periodic Advertising Sync Transfer procedure.

7.7.4 PAwR - LE Periodic Advertising with Responses

7.7.4.1 Basics

PAwR is similar to periodic advertising (PADVB) in several ways:

- PADVB allows application data to be transmitted by one device (the Broadcaster) to one or more receiving devices (the *Observers*), forming a one-to-many communication topology. The same is true of PAwR.
- PAwR and PADVB both use a connectionless communication method.
- Transmission of advertising packets is periodic with a fixed interval and no random perturbation of the schedule in both cases.

- Observers can establish the periodic transmission schedule used by the Broadcaster from AUX_ADV_IND PDUs or by using the Periodic Advertising Sync Transfer (PAST) procedure.

PAwR differs from PADVB as follows:

- PADVB supports the unidirectional communication of data from a Broadcaster to Observers only. PAwR Observers can transmit response packets back to the Broadcaster. PAwR provides a *bidirectional*, connectionless communication mechanism.
- Synchronization information for periodic advertising *without* responses (PADVB) is contained within the SyncInfo field of AUX_ADV_IND PDUs. Synchronization information for periodic advertising *with* responses (PAwR) is contained within the SyncInfo field and in the ACAD field of AUX_ADV_IND PDUs.
- The PADVB Broadcaster schedules transmissions within advertising *events*. The PAwR Broadcaster schedules transmissions in a series of events *and subevents,* and Observers are expected to have synchronized in such a way as to listen during a specific subevent or subevents only.
- The PAwR Broadcaster may use a transmission time slot to send a connection request (AUX_CONNECT_REQ PDU) to a specific device and establish an LE-ACL connection with it. PADVB does not have this capability.
- With periodic advertising without responses (PADVB), application data tends to only change from time to time. PAwR is designed with the expectation that application data will change frequently.
- With PADVB, the same application data is delivered to all Observer devices synchronized to the same advertising set. With PAwR, different data can be delivered to each Observer device or set of Observer devices.

Support for the Periodic Advertising Sync Transfer (PAST) procedure is optional with PADVB but mandatory with PAwR.

7.7.4.2 Channel Use

Channel selection is accomplished using Channel Selection Algorithm #2, and takes place at each periodic advertising subevent (see *7.7.4.3 Scheduling*). Responses to PDUs transmitted in a subevent use the same channel. This includes AUX_SYNC_SUBEVENT_RSP PDUs sent in response to a AUX_SYNC_SUBEVENT_IND PDU and AUX_CONNECT_RSP PDUs which are sent in response to AUX_CONNECT_REQ PDUs.

7.7.4.3 Scheduling

As with other advertising modes, activity takes place in events which in the case of PAwR are known as *Periodic advertising with responses events* (PAwR events). These events occur at fixed intervals, with no random perturbation in scheduling. An event starts every *periodic advertising interval* ms.

Each PAwR event consists of several subevents, and it is during subevents that advertising packets are transmitted. The Host configures the number of subevents per event up to a maximum of 128. A subevent starts every *periodic advertising subevent interval* ms. The Host configures the number of subevents per event and the periodic advertising subevent interval using a Host Controller Interface (HCI) command called HCI_LE_Set_Periodic_Advertising_Parameters V2 (or later).

See *Figure 23 - PAwR events and subevents*.

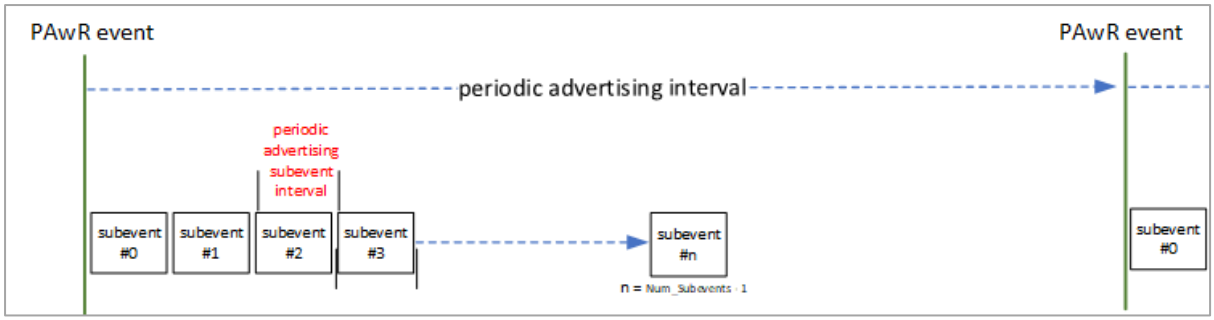


Figure 23 - PAwR events and subevents[6]

In each subevent, the Broadcaster transmits one packet, which usually contains an AUX_SYNC_SUBEVENT_IND PDU but may instead contain an AUX_CONNECT_REQ PDU. After a delay, known as the *Periodic Advertising Response Slot Delay*, a series of time slots are reserved within the same subevent for receiving responses from Observer devices. Responses to AUX_SYNC_SUBEVENT_IND PDUs are sent in AUX_SYNC_SUBEVENT_RSP PDUs. The Host configures the number of response slots required by the HCI command HCI_LE_Set_Periodic_Advertising_Parameters. Figure 24 illustrates the structure of a PAwR subevent.

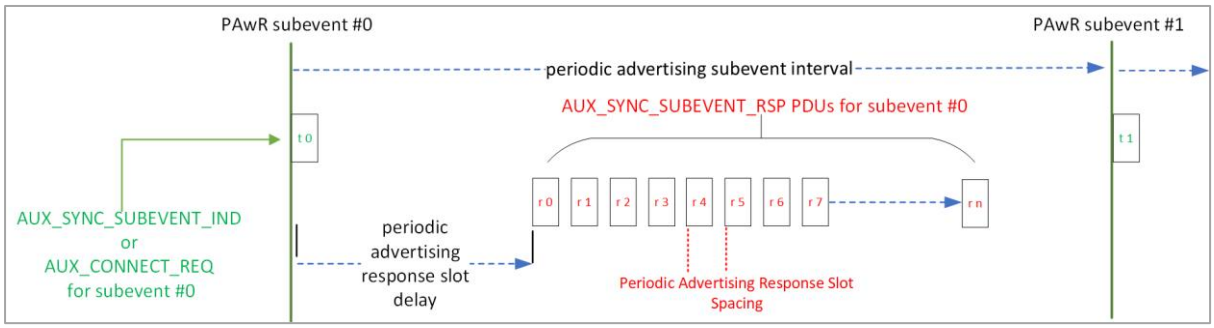


Figure 24 - A PAwR subevent with response slots

7.7.4.4 Synchronization Establishment

General

The process of *synchronizing* provides the Observer device with the information it needs to efficiently scan for and receive relevant packets transmitted by the advertising device. In the case of PAwR, there are three aspects to this:

1. The Observer needs to know how often *periodic advertising with responses event*s will occur and when the next such event will occur. This information is provided in a parameter called the *periodic advertising interval* and a calculated value known as *syncPacketWindowOffset*.

2. The Observer needs information about *subevents,* including how often they occur and how many subevents each *periodic advertising with responses event* accommodates. It also needs to know certain details relating to the time slots within each subevent reserved for the transmission of responses. This information is contained within parameters known as

[6] #nse - 1 means Number of Subevents minus one

Subevent_Interval, Num_Subevents, *Response_Slot_Delay*, *Response_Slot_Spacing*, and *Num_Response_Slots*.

3. Finally, an Observer needs to know which subevent number it should scan for, which particular response slot it should use, and the access address to use in response packets transmitted.

Having acquired the event timing information described in (1) and the subevents information in (2), the Observer has a complete description of the timing parameters and structure of the events and subevents of the PAwR advertising train. But it is only when it has the information in (3) that it can schedule its scanning such that it receives only those packets that are expected to contain data of relevance and can schedule the transmission of response packets.

(1) and (2) are dealt with by the PAwR logical transport, as defined in the Bluetooth Core Specification. There is a choice of two procedures that may be used to obtain this level of synchronization information. The two procedures are covered in this paper in sections *Scanning for Periodic Advertising Synchronization Information* and *Periodic Advertising Sync Transfer (PAST)*.

(3) must be addressed by the application layer and may be defined in an applicable Bluetooth profile specification such as the Electronic Shelf Label (ESL) profile.

Scanning for Periodic Advertising Synchronization Information

PAwR and PADVB each use a similar procedure for acquiring periodic advertising synchronization information by scanning.

With both PAwR and PADVB, an Observer scans for AUX_ADV_IND packets transmitted on the secondary advertising channels. AUX_ADV_IND includes the SyncInfo field, which contains the *periodic advertising interval* value and some data items from which to calculate a variable called *syncPacketWindowOffset*. Having acquired these two values, the Observer can calculate when *periodic advertising with responses event*s will occur, per (1) in *General*.

PAwR also requires information about subevents and response slots, per (2) in *General,* before it can complete the synchronization procedure. This information is to be found in the same AUX_ADV_IND PDU from which the *periodic advertising interval* was obtained but in an advertising data type (AD type) called Periodic Advertising Response Timing Information which itself is to be found in the Additional Controller Advertising Information (ACAD) field of the PDU.

Periodic Advertising Sync Transfer (PAST)

When using the PAST procedure, sometimes the device passing the synchronization parameters over the connection will first acquire it by scanning on behalf of the other device. In the case of PAwR, however, support for PAST is mandatory and so the PAwR Broadcaster can pass the required synchronization data over an LE ACL connection to the Observer. If this approach is taken, no scanning for AUX_ADV_IND PDUs is necessary by either device.

Subevent Synchronization and Response Slot Allocation

Subevent synchronization is concerned with indicating to an Observer device the subevent it should perform scanning for. One or more Observer devices may be synchronized to the same subevent. An individual Observer may be synchronized to receive during one or more subevents.

In addition, for an Observer to be able to send a response PDU, it must have some basis for determining which subevent response slot to use.

Both of these concerns are the responsibility of the application layer.

7.7.5 LE BIS and LE CIS - Isochronous Communication

This section highlights key aspects of isochronous communication. For additional information, the book *Introducing Bluetooth LE Audio by Nick Hunn* is recommended. The book is available in electronic form as a free download from https://www.bluetooth.com/bluetooth-resources/le-audio-book/ . The Bluetooth Core Specification contains full details.

7.7.5.1 Basics

Isochronous communication provides a way of using Bluetooth LE to transfer time-bounded data between devices. It provides a mechanism that allows multiple sink devices, receiving data from the same source at different times to synchronize their processing of that data. LE Audio uses isochronous communication.

When using isochronous communication, data has a time-limited validity period, at the end of which it is said to expire. Expired data which has not yet been transmitted is discarded. This means that devices only ever receive data which is valid with respect to rules regarding its age and acceptable latency which a profile might express.

Data is transferred in *isochronous streams* and streams belong to *isochronous groups*. Devices wait for a period of time to allow all streams that are members of the same group to have the opportunity to deliver related packets before processing received packets at the same time. For example, stereo music might be delivered using two streams, one for the left stereo channel and the other for the right stereo channel. The two streams would be members of the same group and as such, rendering of packets received from the two streams takes place at the same time so that the user hears stereo music as intended.

Two logical transports which use the LE Isochronous physical channel are defined. *Connected Isochronous Streams* (LE CIS or simply *CIS*) use connection-oriented communication and support the bidirectional transfer of data. *Broadcast Isochronous Streams* (LE BIS or simply *BIS*) use connectionless, broadcast communication and provide unidirectional communication of data.

7.7.5.2 Connected Isochronous Streams

7.7.5.2.1 CIS Overview

A single CIS stream provides point-to-point isochronous communication between two connected devices and transfers data in a link layer PDU known as the *CIS PDU*. The LE-CIS (CIS) logical transport is depicted within the overall data transport architecture in Figure 25.

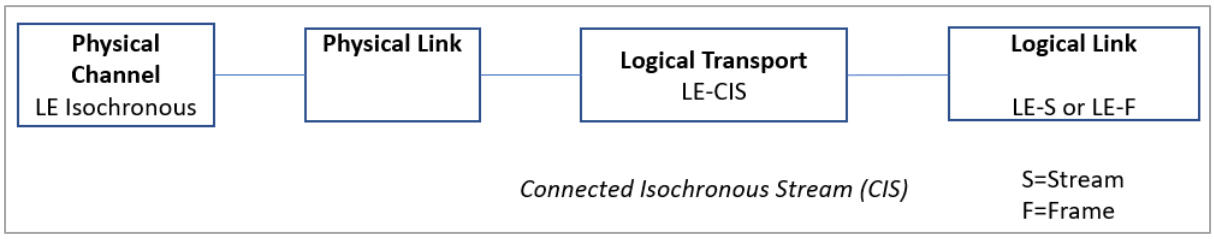


Figure 25 - LE-CIS in the Bluetooth Data Transport Architecture

Two logical links, LE-S and LE-F are defined and provide support for both *unframed* (LE-S) and *framed* (LE-F) data. The use of LE-S vs LE-F is a concern of the Isochronous Adaptation Layer.

A CIS stream uses the LE Isochronous Physical Channel and may use any of the Bluetooth LE PHYs.

Bidirectional communication is supported by a CIS and an acknowledgement protocol is used.

CIS streams are members of groups called Connected Isochronous Groups (CIGs), each of which may contain 1 or more CISes. See Figure 26.

There may be a maximum of 31 CISes per CIG. Multiple CIGs may be created by the Central device. Available airtime and other implementation details will often reduce these limits to lower values however.

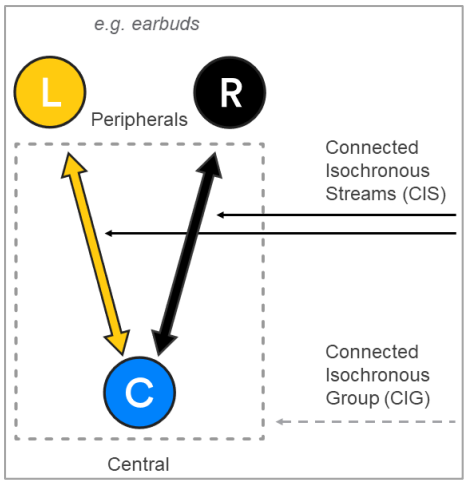


Figure 26 - A CIG containing two CISes

7.7.5.2.2 Channel Use

Connected Isochronous Streams use adaptive frequency hopping with channel selection algorithm #2.

7.7.5.2.3 Scheduling

The scheduling of a CIG and its member CISes is governed by a system of CIG events, CIS events and subevents.

A CIG event signals the start of the scheduling of activity across the CISes that belong to the CIG and this occurs at the *anchor point* of the first CIS in the group. CIG events occur at an interval specified in a parameter called *ISO_Interval*.

Each CIS event is divided into one or more subevents. The number of subevents in use is indicated in a stream parameter called *NSE*. In a connected isochronous stream, during a subevent, the Central transmits (T) once and the Peripheral responds (R) as shown in Figure 27. Subevents are spaced apart by a duration whose value is specified in a CIS parameter called Sub_Interval. Sub_Interval is always set to zero if there is only one subevent per CIS event, otherwise it is at least 400 microseconds but less than the ISO Interval.

Note that the channel is changed at each subevent.

Each CIS may be serviced sequentially during a CIG Event or the subevents of different CISes may be interleaved. An example of a CIG which contains three CISes, each of which is serviced sequentially is shown in Figure 27.

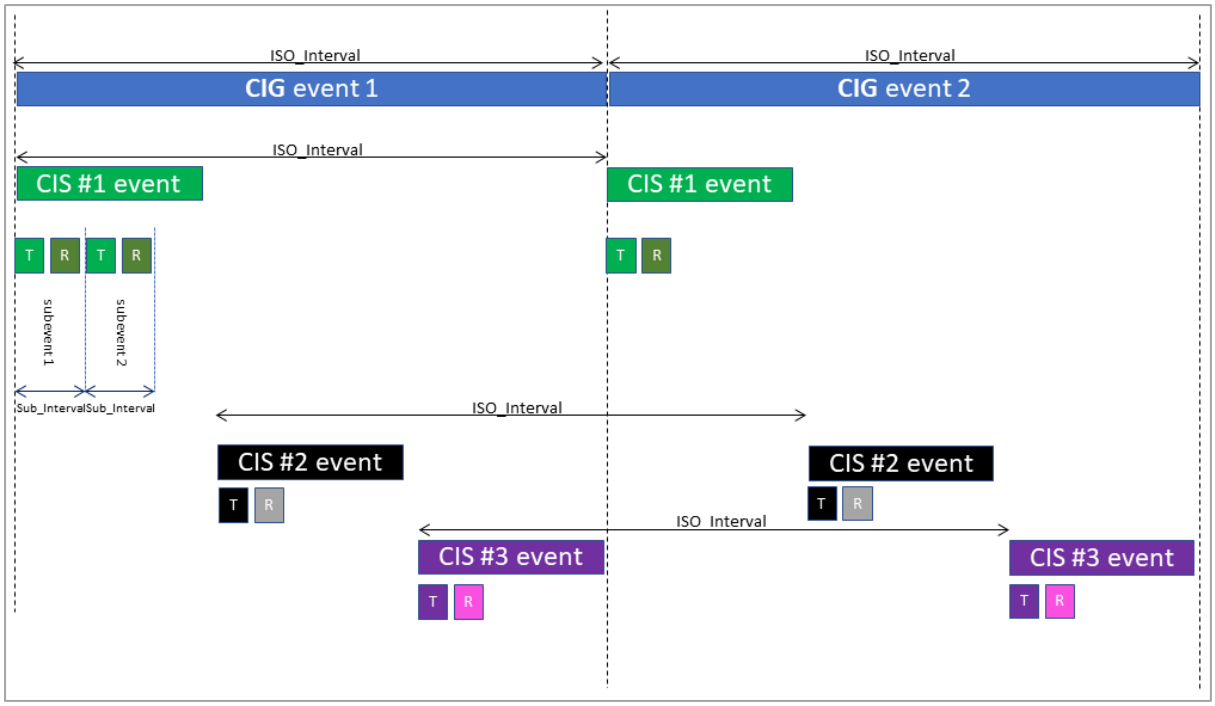


Figure 27 - CIS events and subevents

Each CIS has a number of important parameters other than Number of Subevents (NSE) including Flush Timeout (FT) and Burst Number (BN).

Each payload (e.g. chunk of audio data output by an audio codec such as LC3[7]) is given a maximum number of CIS events in which to be transmitted successfully (indicated by an acknowledgement) and this is specified in the FT parameter. An attempt can be made in each of the subevents of each such CIS event and if not successful within FT events, the packet is *flushed* (discarded). It is sometimes the case that multiple PDUs containing distinct data (i.e., payloads) are available at the same time and a CIS allows multiple different PDUs to be transmitted during the same CIS event. The number of different PDUs which may be serviced at each CIS event is specified in the Burst Number (BN) parameter.

7.7.5.2.4 Processing Synchronization

A CIG has an associated timing parameter called **CIG**_Sync_Delay. CISes in the same CIG each have a timing parameter called **CIS**_Sync_Delay and this is used in the synchronization of isochronous data processing (typically, audio rendering) by receivers across all of the streams in the group. Receivers wait for the time indicated in this parameter before rendering received data.

[7] The Low Complexity Communications codec used by LE Audio

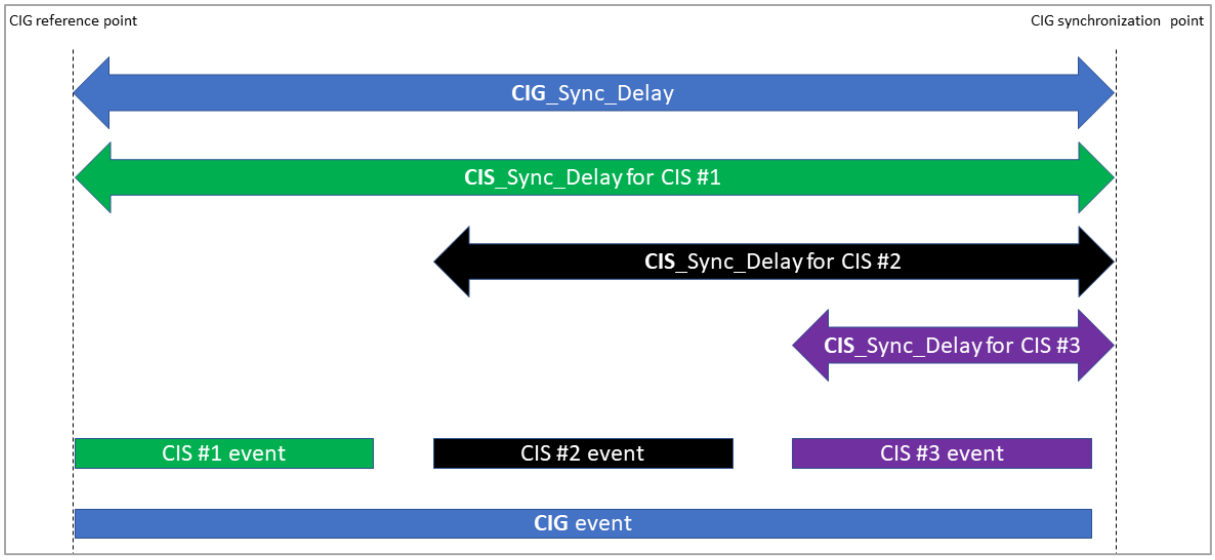


Figure 28 - Synchronized rendering of CIS data in a CIG

As depicted in Figure 28, each stream has a different CIS_Sync_Delay value. For the first CIS stream in the CIG, it is set to the group level parameter CIG_Sync_Delay. CIS_Sync_Delay is set to a progressively lower value for each of the other streams in the group. This means that a device receiving from a stream that was serviced earlier in the group has to wait longer before rendering the content of the received packet than devices receiving packets transmitted later in the group's CIS event processing. Higher layer specifications such as profiles may stipulate the use of a further presentation delay to use in the calculation of the time at which data should be rendered to allow local processing delays to be accounted for. The net of this tiered delay system is that each sink device will process the received data at the same time.

7.7.5.2.5 CIS Stream Creation

Establishing a connected isochronous stream first requires an ACL connection to be created. This connection serves two purposes. First it allows link layer control PDUs to be exchanged. Secondly it provides a timing reference point against which to schedule CIS events once the stream has been established.

The Central device always initiates the procedure to create a CIS. It does so by sending a link layer control PDU called the LL_CIS_REQ PDU. All being well, the Peripheral replies with a LL_CIS_RSP PDU and the stream is said to have been established when the Central then sends a LL_CIS_IND PDU. This PDU contains important parameters which determine the timing of CIS events and the delay to apply before rendering. Specifically, CIS_Offset provides an offset in microseconds between the ACL anchor point (the time at which the first packet is sent in a connection event) and the first CIS event for the stream. CIG_Sync_Delay contains the overall CIG synchronization delay value in microseconds and CIS_Sync_Delay contains the synchronization delay value to be used by this stream.

After the stream has been created, it runs independently of and alongside the ACL connection which was used to create it. If however the ACL connection is closed, the associated CIS must also be terminated.

7.7.5.2.6 CIS Encryption

A link used by a CIS may be encrypted if the two peer devices have been paired.

7.7.5.3 Broadcast Isochronous Streams

7.7.5.3.1 BIS Overview

A BIS stream provides broadcast isochronous communication between one transmitter (source) and many receiver (sink) devices. Data is transmitted in link layer PDUs known as the *BIS Data PDU*. Control information is transmitted in *BIS Control PDUs*.

The LE-BIS (BIS) logical transport is depicted within the overall data transport architecture in Figure 29.

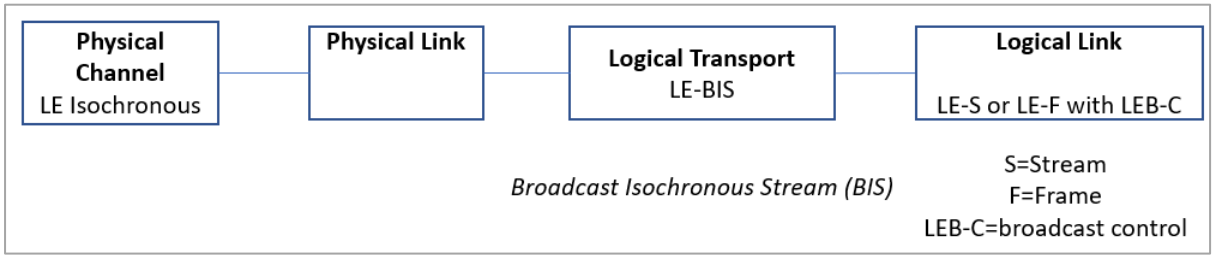


Figure 29 - LE-BIS in the Bluetooth Data Transport Architecture

Data broadcast over a BIS may be framed or unframed with logical link types LE-S and LE-F defined accordingly. The LEB-C logical link carries control information.

A BIS stream uses the LE Isochronous Physical Channel and may use any of the Bluetooth LE PHYs.

BIS streams are members of groups called Broadcast Isochronous Groups (BIG), each of which may contain 1 or more BISes. See Figure 30.

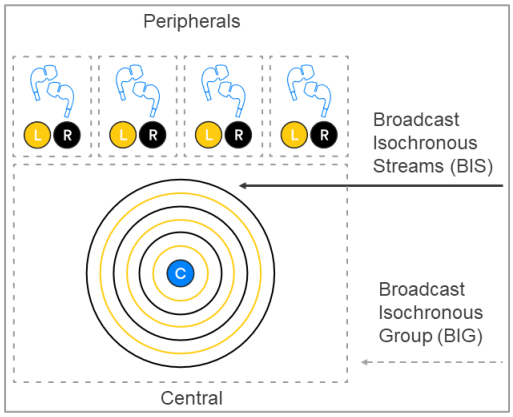


Figure 30 - A BIG containing two BISes

There may be a maximum of 31 BISes per BIG. Multiple BIGs may be created by the Central device. Available airtime and other implementation details will often reduce these limits to lower values however.

Unidirectional communication only is supported by a BIS.

In contrast to a CIS, a BIS does not incorporate an acknowledgement protocol. This makes the BIS transport inherently unreliable. However, to counter this a system of *unconditional packet retransmission* is used. A BIS has no requirement to reserve slots for Peripheral responses (as is the case with CIS) since communication is unidirectional. Therefore, twice as many subevents may be scheduled for transmissions during a given amount of airtime so there is a greater opportunity for these reliability-enhancing retransmissions. Furthermore, since retransmissions are sent in distinct

subevents they are transmitted on different channels. Selected channels must be at least 6 MHz from the last transmission, and this helps mitigate potential packet loss due to interference on a particular channel.

7.7.5.3.2 Channel Use

Broadcast Isochronous Streams use adaptive frequency hopping with channel selection algorithm #2.

7.7.5.3.3 Scheduling

The scheduling of a BIG and its member BISes is governed by a system of BIG events, BIS events and subevents. In addition, a special *control subevent* is defined for the transmission of control PDUs relating to the entire BIG.

A BIG event signals the start of the scheduling of activity across the BISes that belong to the BIG. BIS events start at intervals which are a multiple of the value specified in a BIG a parameter called *BIS_Spacing* from the start of the BIG (known as the *BIG anchor point*).

Each BIS event is divided into one or more subevents. The number of subevents in use is indicated in a stream parameter called NSE. During a subevent, the broadcaster transmits a single packet. Communication is unidirectional and there is no requirement to receive packets. Subevents are spaced apart by a duration whose value is specified in a BIG parameter called Sub_Interval.

Per connected isochronous groups, the scheduling of BIS events within a BIG may either be sequential or interleaved.

A BIG event may include a *control subevent* which is always scheduled as the final subevent in the BIG.

Note that the channel is changed at each subevent.

Figure 28 shows an example of the BIG and BIS events and subevents, scheduled in the sequential arrangement. Note that a BIG control subevent (designated Tc) is transmitted at the end of BIG event #1.

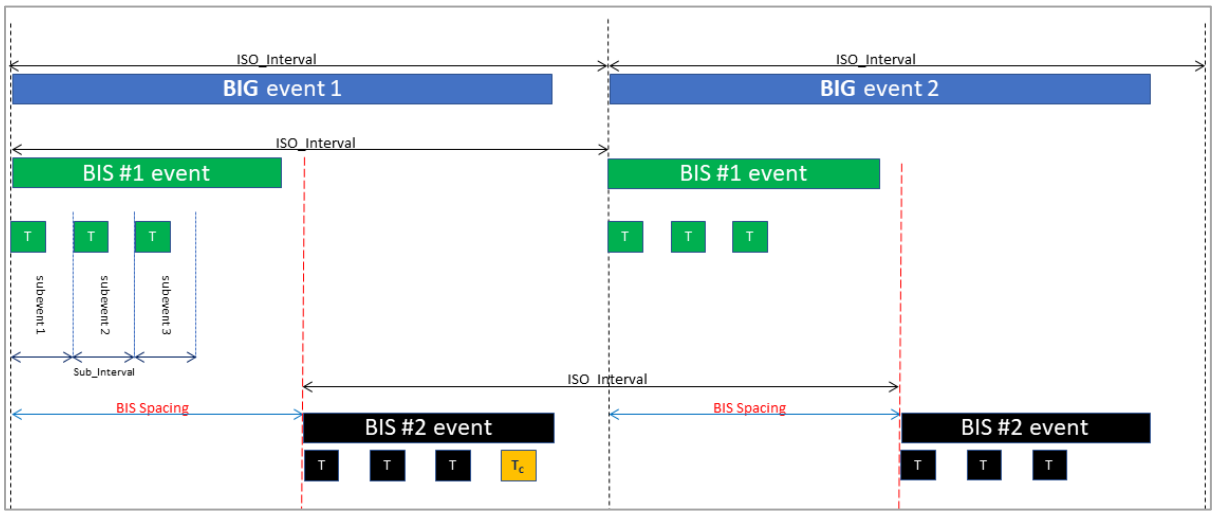


Figure 31 - BIG/BIS event scheduling

7.7.5.3.4 Processing Synchronization

Synchronized processing of data across broadcast isochronous streams in a BIG is achieved in a similar manner to the approach used in connected isochronous communication. Receivers possess information about the BIG and its overall parameters and know which stream(s) they have opted to receive. The timing parameters of a BIG apply uniformly to all streams. Using the overall BIG_Sync_Delay value and the BIS_Spacing parameter, receivers are able to calculate how long to wait for before processing received data such that this occurs in sync with other streams.

7.7.5.3.5 BIS Stream Creation

For a device to be able to receive packets broadcast within a BIS and to render or process the content of such packets at the same time as other devices receiving other streams that are members of the same BIG, the device must first discover the BIG and parameters which define it such as the number of streams it contains, the spacing between events relating to each stream and between subevents and timing offset information from which to calculate timing anchor points. In support of this, broadcasters use periodic advertising to communicate the required parameters. A composite field known as BIGInfo is broadcast in AUX_SYNC_IND PDUs within the ACAD (Additional Controller Advertising Data) field and contains the data required.

There are two ways in which BIGInfo may be received. In the first case, the receiver must synchronize directly to the periodic advertising train (see 7.7.3.1 Basics) using the defined procedure, receive the AUX_SYNC_IND PDU and extract BIGInfo from within ACAD. Scanning for and synchronizing with a periodic advertising train can be an expensive procedure in terms of power consumption however. So in the second case, a device may delegate the act of discovering and synchronizing with the periodic advertising train to another device, typically one which has greater power resources. Having acquired BIGInfo, the device to which scanning was delegated then passes this information over a more efficient ACL connection to the device wishing to receive the broadcast isochronous stream. This is accomplished using a procedure called Periodic Advertising Sync Transfer (PAST).

7.7.5.3.6 BIG Encryption

A BIG may be encrypted. This does not require devices receiving its BISes to have been paired with the broadcasting device. Instead, a *Broadcast Code* parameter which is used in the derivation of an encryption key must be distributed. This may be performed out of band or by following procedures described in higher level profiles.

7.7.5.4 Retransmissions and Reliability

Reliability may be enhanced using retransmissions of identical packets within successions of subevents on either BIS or CIS streams. In the case of BIS, retransmissions are unconditional whereas with CIS they occur when the Peripheral has not acknowledged a transmission.

In the case of BIS, because there is no requirement to reserve slots for Peripheral responses (as is the case with CIS), twice as many subevents may be scheduled for transmissions during a given amount of air time so there is a greater opportunity for reliability-enhancing retransmissions.

Retransmissions, due to their occupying distinct subevents, are transmitted on different channels and selected channels must be at least 6 MHz from the last transmission. This helps mitigate potential packet loss due to interference on a particular channel.

7.7.5.4 LE Audio

LE Isochronous communication was primarily designed for use in audio products and systems. It provides the means by which audio, delivered from a source to multiple sinks, can be rendered at the same time, for properly synchronized playback.

8. The Isochronous Adaptation Layer

8.1 Basics

The purpose of the Isochronous Adaptation Layer (ISOAL) is largely to address a potential issue that can affect both connected and broadcast isochronous communication involving audio devices. It may find application in other uses of isochronous communication.

Chapter 5.2 of Nick Hunn's book *Introducing Bluetooth LE Audio* covers this topic well and is recommended reading.

8.1.1 Audio Sampling 101

Digital audio works by sampling an analogue audio signal and applying a codec to the sampled audio to compress and otherwise process the digital sample data before storing or in the case of Bluetooth LE audio, transmitting it. On reading or receiving encoded digital audio data, the process is reversed with a codec used to decode the data, producing a series of digital samples which are then used to (approximately) recreate the original analogue audio. Figure 32 illustrates the steps involved in sampling, encoding and transmitting an audio signal and the reverse steps involved in receiving encoded audio data, decoding it and finally, rendering it.

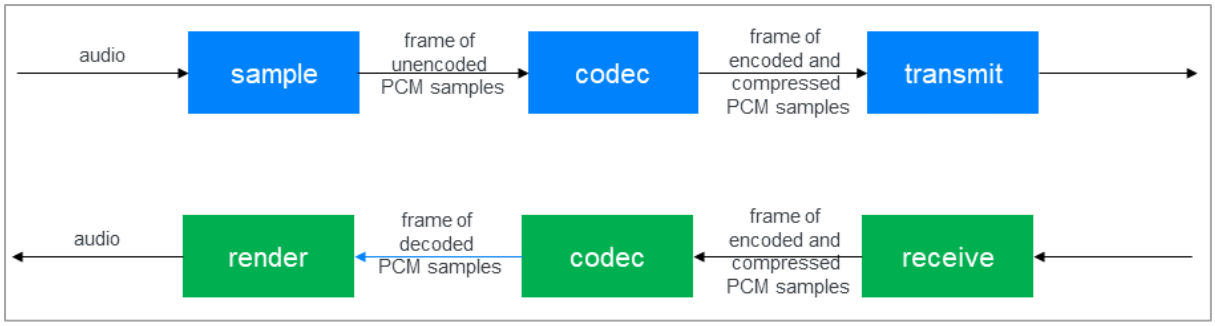


Figure 32 - Audio processing steps

One of the key goals of an audio codec is to reduce the size of the audio data so that it can be efficiently transmitted over a link which has limited (and precious!) bandwidth. Sampling involves measuring and recording the amplitude of a signal at regular intervals. The frequency with which a sample is recorded is known as the *sample rate*. The vertical lines in Figure 33 represent samples of the continuously variable audio signal, represented by the curve. That series of samples creates an *approximate representation* of the original analogue signal. The more frequently samples are taken (i.e., the higher the sample rate) the closer that approximation is to the original and on reversing the process to recreate the original audio, the better the results and the perceived quality.

A further dimension of sampling is the *bit depth*. The amplitude of the signal when sampled needs to be represented by an integer value. One approach is to divide the range of possible amplitude values into 256 discreet amplitude bands and represent each by a number between 0 and 255. In this scheme, a single sample needs only one byte (8 bits) to record the band that the sampled amplitude value falls within and the bit depth is therefore said to be 8 bits. Dividing the possible range of amplitudes into more discreet bands provides a more fine-grained system for representing sample values and so potentially better-quality results. A bit depth of 24 bits for example, allows the range of possible amplitude values to be represented by an integer in the range 0 to 16,777,215. Each

sample requires 3 bytes however and hence three times as much data is produced by sampling with this increased bit depth.

Digital sampling can produce large amounts of data. Consider a three-minute song, sampled at a rate of 44.1 kHz (CD quality) and with a bit depth of 24. If we do the math, we find that the amount of raw sampled data required to approximately represent the whole song in digital form is nearly 24 megabytes. This is not something we'd worry about if all we wanted to do is to store the data on the 4-terabyte hard drive of a computer but when transmitting that data over any communication link with limited bandwidth, it's an issue. That's where codecs come in.

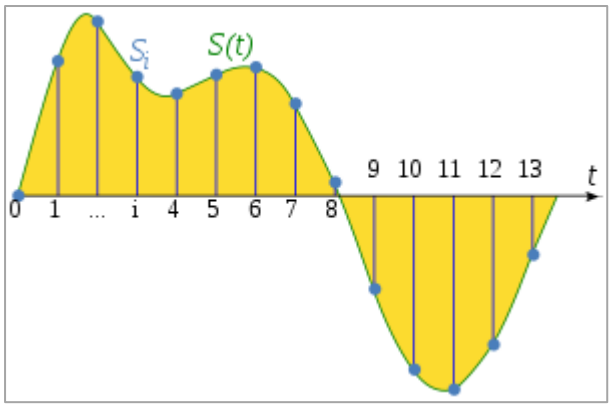


Figure 33 - Discrete samples representing a continuous analogue signal

8.1.2 Codecs and Frames

A codec like LC3 which is used by Bluetooth LE audio might compress raw digital sample data to less than 25% of the original size (but note that actual results depend heavily on the original audio content). This is a substantial saving.

Codecs generally work by recognizing and exploiting patterns found in a series of consecutive samples. To give a very simple example and solely to illustrate this principle, if the data set contained a consecutive series of 100 sample values all with the same value of say, 50 then assuming a bit depth of 8 bits, a codec might represent this as two bytes containing [100,50] rather than the original series of 100 bytes containing the list of individual samples [50,50,50...,50]. Clearly, for a codec to work, it needs a whole series of samples to analyze and encode as opposed to working on one sample at a time (not many patterns to be found in a single sample!).

A collection of samples analyzed at one time by a codec is known as a *frame*. Frames have a fixed duration, normally measured in milliseconds and contain a number of samples determined by the sample rate. For example, a 10ms frame contains 4410 samples if the sample rate is 44.1 KHz.

Different audio products may use different frame durations. 10ms and 7.5ms are common. When one device is producing audio (the *source*) with one frame duration, and another is consuming it (the *sink*) but uses a different frame duration then there's a problem that needs to be solved. That's where ISOAL comes in.

8.2 Framed vs Unframed

When devices use isochronous communication, the frame durations used by the transmitting device and a receiving device do not need to be the same. This leads to two possible situations:

1. The frame duration used by the first device is an exact multiple of the frame duration used by the other device.
2. The frame duration used by the first device **is not** an exact multiple of the frame duration used by the other device.

In the first situation, the relationship between the larger frame duration and the smaller is simple and transforming data between the two is a straightforward operation. Data sent in the payload of one or more required link layer PDUs is said to be *unframed* and includes no additional data to support the adaptation of the frame duration between the two requirements.

In the second, link layer PDUs may contain parts of the larger payload together with short header fields that indicate whether a part is the start of a frame, a continuation or the end of a frame. Data formatted like this is said to be *framed*.

Both Connected Isochronous PDUs and Broadcast Isochronous PDUs (defined by the link layer) contain a field called *LLID*. LLID indicates whether the payload of the link layer PDU contains framed or unframed data. The ISOAL applies different processes to data it receives from the link layer depending on whether data is framed or not. Similarly, whether framing is required or not affects ISOAL processing of data it receives in upper layer SDUs and passes to the link layer for transmission in link layer ISO PDUs.

8.3 Fragmentation and Recombination

If data is *unframed* then *recombination* creates a single service data unit (SDU) from a series of one or more *fragments* contained within the payload of one or more link layer PDUs. The ISOAL then passes the SDU to the upper layer. This is shown in Figure 34.

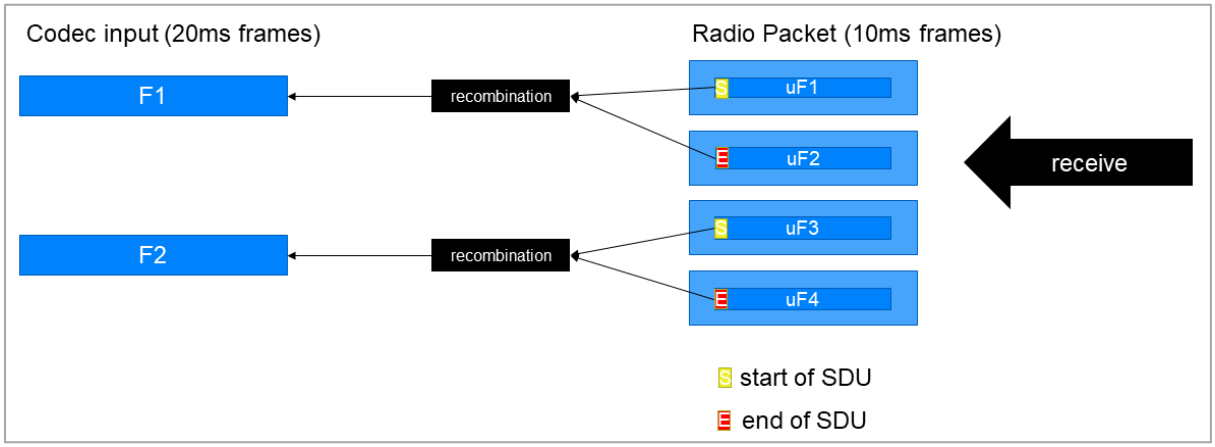


Figure 34 - Recombination of unframed ISO PDUs

When upper layer SDUs need to be split into smaller payloads for transmission in link layer PDUs and framing is not required, the process is called *fragmentation*. This is shown in Figure 35.

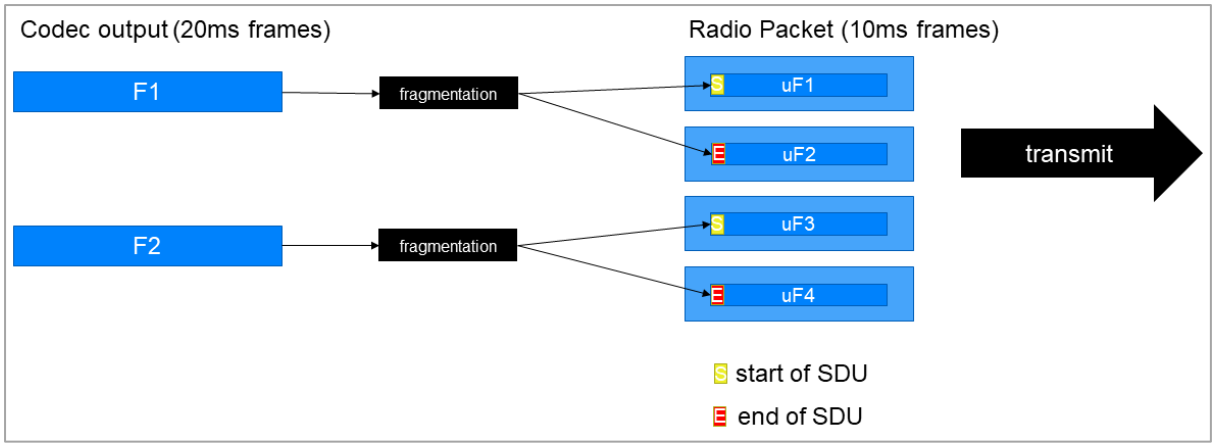


Figure 35 - Fragmentation creating unframed ISO PDUs

Unframed PDUs have a header which includes fields that indicate that the data which follows it is either the start of an SDU, the continuation of the previous SDU or the end of this SDU. PDUs only ever contain a single fragment of an unframed SDU.

8.4 Segmentation and Reassembly

If data is *framed* then *reassembly* creates a single service data unit (SDU) from a series of one or more *segments* contained within the payload of one or more link layer PDUs. The ISOAL then passes the SDU to the upper layer. This is shown in Figure 36.

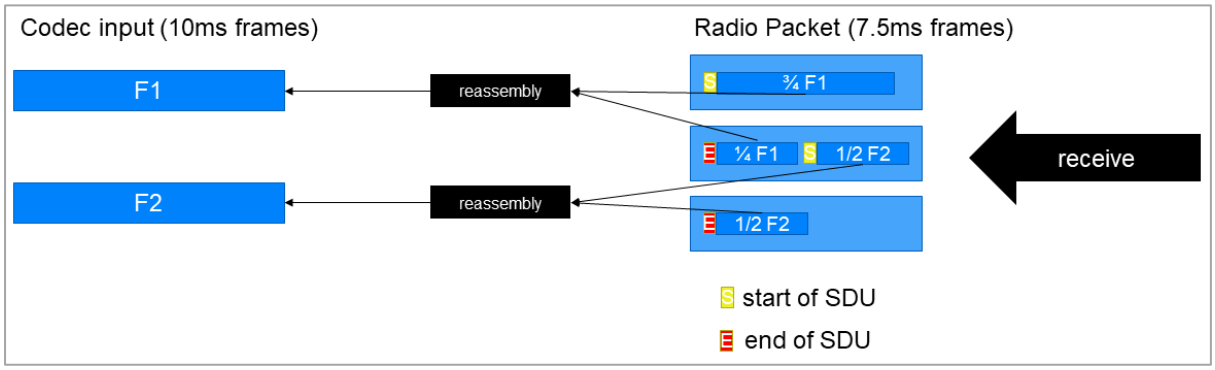


Figure 36 - Reassembly of framed ISO PDUs

When upper layer SDUs need to be split into smaller payloads for transmission in link layer PDUs and framing is required, the process is called *segmentation*. This is shown in Figure 37.

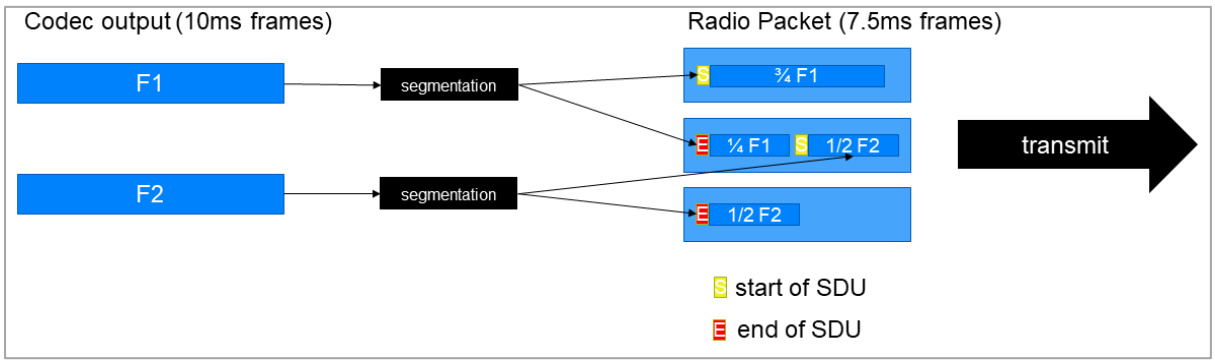


Figure 37 - Segmentation creating framed ISO PDUs

Segments within framed PDUs have a header which includes fields that indicate that the data which follows it is either the start of an SDU, the continuation of the previous SDU or the end of this SDU and timing offset information. PDUs may contain multiple fragments of a framed SDU.

9. The Host Controller Interface

9.1 Basics

The Host Controller Interface (HCI) defines a standardized interface via which a host can issue commands to the controller and a controller can communicate with the host. The specification is split into several parts, the first defining the interface in functional terms only, with no consideration regarding specific implementation mechanisms and the other parts defining how HCI can be implemented when using one of four possible physical transports.

HCI is used by both Bluetooth LE and Bluetooth BR/EDR.

9.2 The HCI Functional Specification

The functional interface is defined in terms of *commands* and *events.* These are essentially messages which may be exchanged between host and controller. Commands are sent by the host to the controller and events from the controller to the host. An event may be a response to a command, or it may be sent by an unsolicited message. See Figure 38.

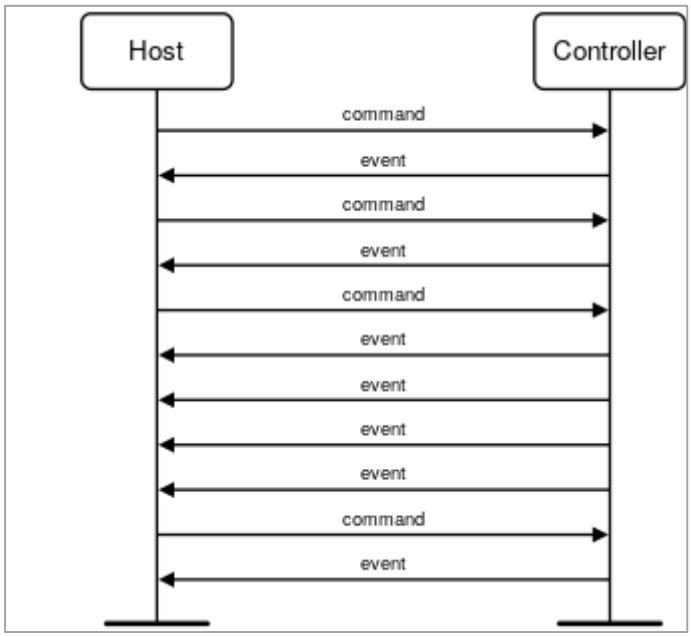


Figure 38 - HCI commands and Events

9.3 HCI Transports

The four HCI transport types are:

1. UART
2. USB
3. Secure Digital (SD)
4. Three-wire UART

9.3.1 UART Transport

The HCI UART transport may be used to implement HCI communication using UART when both host and controller are connected using UART on the same printed circuit board. A protocol based on 5

packet types is defined. These are the HCI Command packet, HCI ACL Data packet, HCI Synchronous Data packet, HCI Event packet and the HCI ISO Data packet.

RS232 configuration requirements are specified. In summary, the UART transport uses 8 data bits, no parity, 1 stop bit and RTS/CTS flow control.

9.3.2 USB Transport

USB may be used as an HCI transport in two ways. The Bluetooth controller may be implemented within a USB dongle. Alternatively, USB can be used internally within a product to link host and controller implementations.

The USB standard defines buffers into which data may be sent or from which it may be received called *endpoints*. The HCI USB transport specification indicates the endpoints that are expected to exist and their required or suggested properties.

9.3.3 Secure Digital

A protocol for use with the HCI SD transport is defined and owned by the Secure Digital Association (SDA). The Bluetooth core specification's section on the HCI SD transport consists largely of references to external specifications owned by the SDA together with an overview of the communication architecture.

9.3.4 Three-wire UART

The Three-wire UART HCI transport specification describes an architecture and protocol for communicating HCI commands and events between two three-wire-connected UARTs. It deals with reliability, data integrity, link establishment, power management and hardware configuration.

9.4 HCI Examples

A number of examples which show the HCI functional interface in action follow.

9.4.1 Connectionless AoA/AoD

Figure 39 shows the exchange of HCI commands and events during the configuration of the controller by the host to prepare for the transmission of packets that support direction finding using either the angle of arrival (AoA) or angle of departure (AoD) techniques.

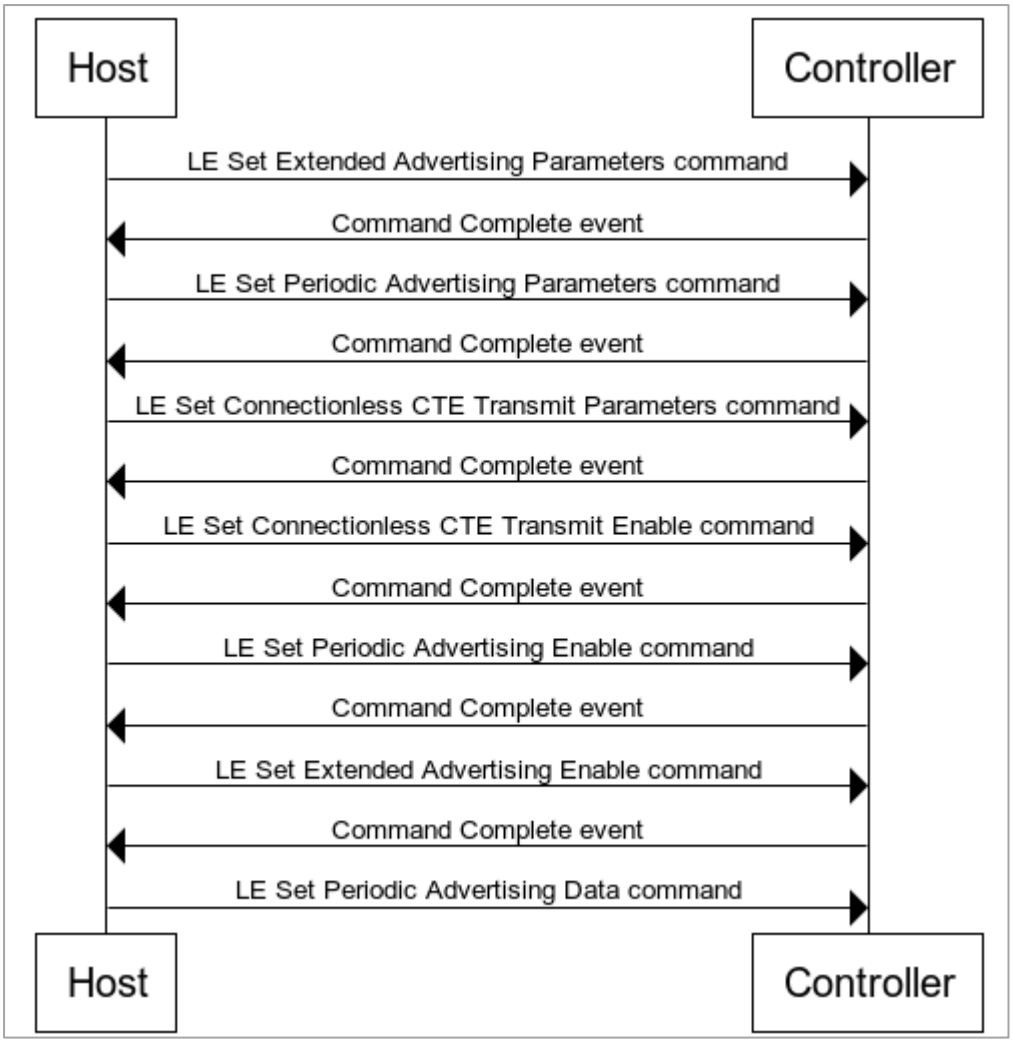


Figure 39 - Connectionless AoA/AoD controller configuration

9.4.2 LE Path Loss Monitoring

LE path loss monitoring is part of the LE power control feature. A device wishing to manage the transmission power used by a connected peer device to keep it within the optimal power range for the receiver might use this feature.

Figure 40 shows a host sending the HCI commands necessary to configure and then enable LE path loss monitoring. After monitoring has been enabled, the controller sends LE Path Loss Threshold events containing path loss data to the host.

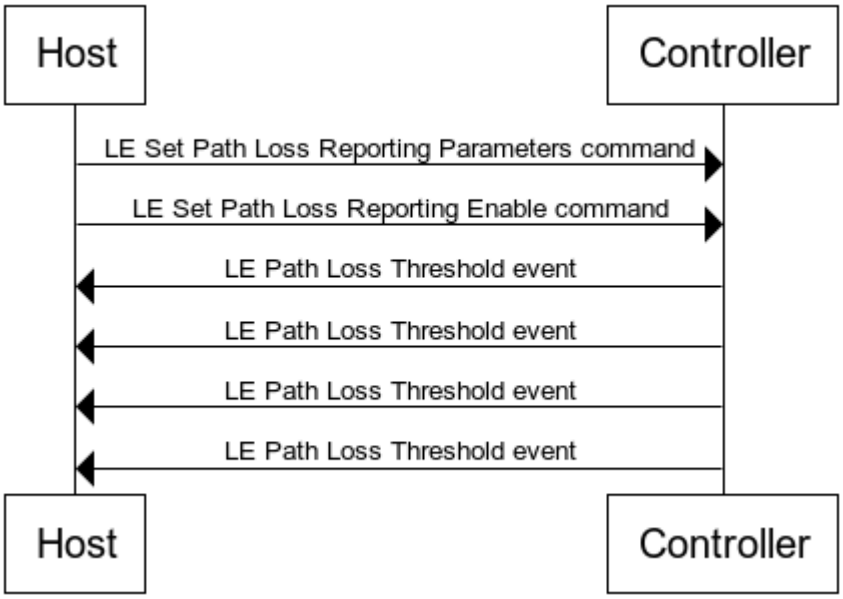


Figure 40 - LE Path Loss Monitoring

9.4.3 Active Scanning

Active scanning is supported by Peripherals which utilise legacy advertising and need to communicate more data than can be contained within a single advertising packet. See section *13.3 Discovery* for more on this topic.

Figure 41 shows the host component in a device (*Device A*) configuring its controller to perform active scanning and then in a separate command, initiating scanning by enabling it. As scannable advertising packets such as ADV_IND are received from another device by the link layer of Device A, due to it having been configured to perform active scanning, it responds by sending a SCAN_REQ PDU and receiving a SCAN_RSP PDU back from the other device. The content of the original advertising packet and the scan response are then passed up to Device A's Host in a series of HCI LE Advertising Report events.

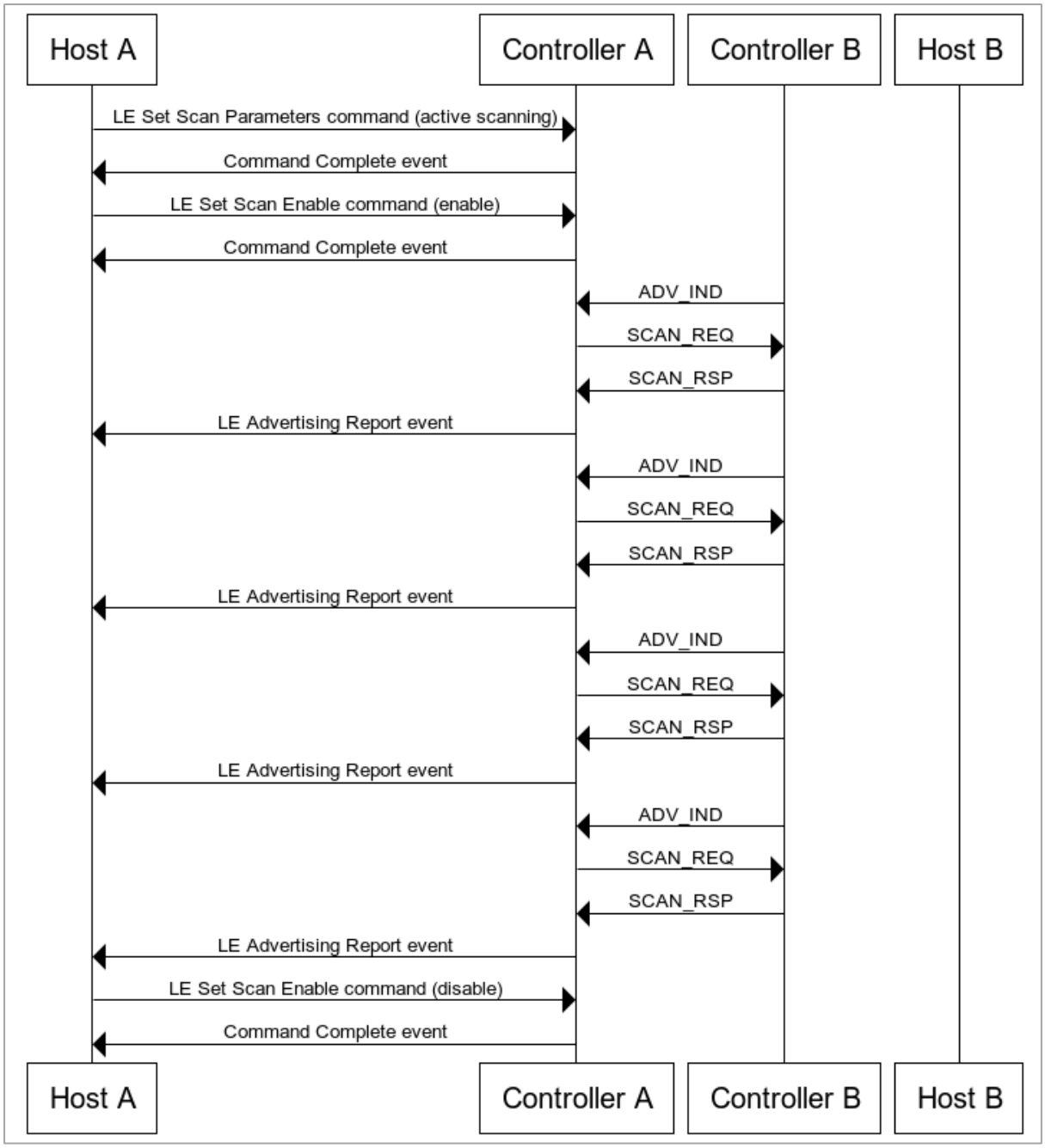


Figure 41 - Active Scanning

10. The Logical Link Control and Adaptation Protocol

10.1 Basics

The Logical Link Control and Adaptation Protocol (L2CAP) is responsible for protocol multiplexing, flow control, and segmentation and reassembly of service data units (SDUs).

L2CAP uses the concept of *channel* to separate sequences of packets passing between layers of the stack. Fixed channels require no set-up, are immediately available and are associated with a particular higher layer protocol. Channels may also be dynamically created and associated with a protocol through a specified Protocol Service Multiplexer (PSM) value.

Figure 42 illustrates the primary L2CAP functions.

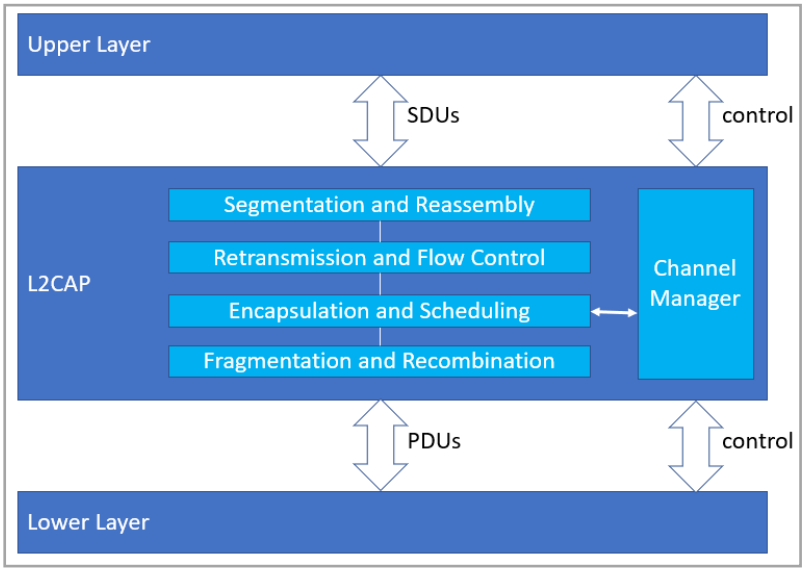


Figure 42 - L2CAP primary functions

10.2 L2CAP and Protocol Multiplexing

Sitting above L2CAP in the stack are layers which use distinct protocols such as the attribute protocol (ATT) and the security manager protocol (SMP). L2CAP protocol multiplexing makes sure that SDUs get passed up the stack into the appropriate layer for processing.

When an L2CAP channel is handling the attribute protocol, it either uses a fixed channel which is reserved for ATT and in this case is said to be acting as an *unenhanced ATT bearer* or it uses a series of one or more dynamic channels, each acting as an *enhanced ATT bearer*. The unenhanced ATT bearer supports ATT transactions executed in sequence, one at a time. The enhanced ATT bearer supports parallel ATT transactions which are executed sequentially within parallel L2CAP channels. See 11. The Attribute Protocol for further details on this.

10.3 L2CAP and Flow Control

Flow control is concerned with making sure that the rate at which packets are produced by one layer of a stack does not exceed the rate at which a layer in the same stack or on a remote device processes those packets. Without flow control, there is a risk of issues such as buffer overflow.

Credit based flow control is one of many possible approaches to flow control. In outline, it works as follows:

- The transmitting device knows the capacity of the receiving device in terms of the number of PDUs it can handle without losing data (e.g., through its buffer overflowing). It acquires this capacity information through configuration or via an exchange made between the two devices before data transfer starts.
- The transmitter sets a counter to the receiver capacity limit. Every time a PDU is sent by the transmitter, the counter is decremented. When the counter value reaches zero, the transmitter knows the receiver is at full capacity and so stops sending further PDUs temporarily while the receiver processes its backlog.
- After the receiver reads and processes one or more PDUs from its buffer, it sends back a corresponding number of credits to the transmitter which uses it to increment its counter. With the counter at a non-zero value, the transmitter may continue to send further PDUs.

L2CAP defines several modes of operation, largely concerned with flow control.

For example, ATT over an L2CAP unenhanced ATT bearer uses Basic L2CAP Mode, which provides no flow control. This renders ATT unreliable and applications must accommodate the possibility that transmitted ATT PDUs may be lost by the receiving device. In the case of unacknowledged PDUs such as notifications, the transmitting device will know whether or not the PDU was received by the stack on the remote device, thanks to link layer acknowledgements but has no way of knowing whether the PDU was successfully delivered to the receiving application at the top of the stack.

ATT over an L2CAP enhanced ATT bearer uses Enhanced Credit Based Flow Control Mode, which provides flow control. EATT may therefore be regarded as reliable.

10.4 L2CAP Segmentation and Reassembly

Layers both above and below L2CAP are subject to a Maximum Transmission Unit (MTU) size which specifies the largest size a PDU of the type created by that layer is allowed to be. For example, the ATT_MTU parameters defines the maximum size an ATT PDU may be.

L2CAP itself and the layers above or below it in the stack may have different MTU sizes and consequently, it may be necessary to split some PDUs/SDUs into a series of smaller parts which the adjacent layer can handle or conversely, to reassemble a series of related, smaller parts into full PDUs/SDUs. Those processes, as applied by L2CAP with respect to upper layers is called *segmentation and reassembly* while the equivalent processes as they relate to L2CAP and its relationship with lower layers is called *fragmentation and recombination*.

11. The Attribute Protocol

11.1 Basics

The attribute protocol (ATT) is used by two devices, one acting in the role of *client* and the other in the role of *server*. The server exposes a series of composite data items known as *attributes*. Attributes are organized by the server in an indexed list called the *attribute table*.

Each attribute contains a handle, a Universally Unique Identifier (UUID), a value and a set of permissions.

- The handle is a unique index value with which an ATT client can reference a specific entry in the attribute table.
- The UUID identifies the attribute's type.
- The permissions field is a set of flags which indicate whether read, write or both forms of access are permitted and any other security conditions which must be met for access to be allowed.
- The Bluetooth SIG study guide *Understanding Security in Bluetooth LE* has more information on attribute permissions.
- The attribute value field is a byte array contain the attribute's value. Interpretation of the byte array both in terms of data types and semantics is the concern of higher layers of the stack.

The Generic Attribute Profile (GATT) defines how attributes may represent higher level constructs known as services, characteristics and descriptors. Typically, a group of attributes in a contiguous handle value range are required to represent more complex types such as these and the attribute protocol supports working with groups of attributes identified by a handle value range for this reason. See section 12. The Generic Attribute Profile for more information on this.

ATT is used by an ATT client to discover details of the attribute table in an ATT server, including the handle values of attributes or attribute types of interest. When handle values are known, they can be used with some PDU types to identify and then act upon specific attributes in the table. For example, the ATT_READ_BY_GROUP_TYPE_REQ PDU can be used to find the handle and UUID of all attributes that are part of the definition of a *primary service*. A shorter way of expressing this is that the ATT_READ_BY_GROUP_TYPE_REQ PDU can be used to find all of the GATT primary services defined by the attribute table, for example.

When using PDUs like ATT_READ_BY_GROUP_TYPE_REQ that support discovery operations, a handle range is specified and indicates the subset of the entries in the attribute table to be searched (and this may be the whole table) and an attribute type to look for. Figure 43 illustrates this process with all primary services being sought and the response indicating the range of handle values containing attributes that relate to the discovered primary service.

Opcode	Read By Group Type Request	0x10
Starting Handle	1	0x0001
Ending Handle	Max Handle	0xFFFF
Attribute Group Type	Primary Service	
Opcode	Read By Group Type Response	0x11
Pair Length	6	0x06
Triple (1)		
Attribute Handle	1	0x0001
End Group Handle	8	0x0008
UUID	Generic Attribute	0x1801
Triple (2)		
Attribute Handle	9	0x0009
End Group Handle	15	0x000F
UUID	Generic Access	0x1800
Triple (3)		
Attribute Handle	16	0x0010
End Group Handle	20	0x0014
UUID	Device Information	0x180A

Figure 43 - Read By Group Type Request and Response

ATT is one of the primary mechanisms by which applications in connected Bluetooth LE devices interact with each other, using the PDUs that the protocol defines, and procedures defined in higher level specifications, such as the Generic Attribute Profile (GATT).

Two variants of ATT are defined, the basic ATT and the newer Enhanced Attribute Protocol (EATT).

ATT may be used by both Bluetooth LE or Bluetooth BR/EDR. In this document only ATT used with Bluetooth LE is considered.

11.2 ATT PDUs

The Attribute Protocol defines 31 distinct PDUs each of which are based upon one of six broad methods.

11.2.1 Commands

An ATT command PDU is sent by a client to a server with no response from the server invoked. An example of a command is the ATT_WRITE_CMD shown in Figure 44.

Figure 44 - ATT_WRITE_CMD

11.2.2 Requests and Responses

An ATT request PDU is sent by a client to a server. The server is expected to reply with a response PDU of a corresponding type or with an error response PDU (ATT_ERROR_RSP) within 30 seconds. Failure to respond within 30 seconds constitutes a time out.

An example of a request / response PDU pairing is the ATT_WRITE_REQ and ATT_WRITE_RSP PDUs shown in Figure 45.

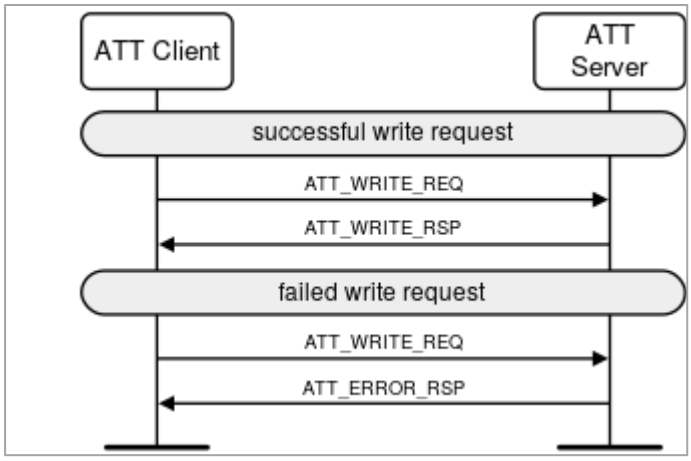


Figure 45 - ATT_WRITE_REQ

11.2.3 Notifications

Notifications are unsolicited PDUs of type ATT_HANDLE_VALUE_NTF that are sent by a server to a client. No reply PDU is defined. See Figure 46.

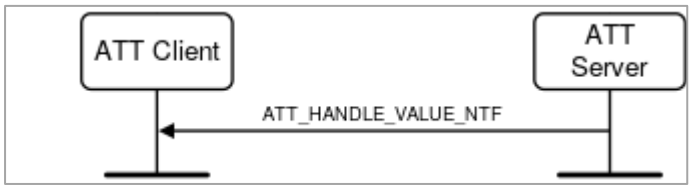


Figure 46 - ATT_HANDLE_VALUE_NTF

11.2.4 Indications and Confirmations

An ATT indication PDU is sent by a server to a client. The client is expected to reply with a confirmation PDU of a corresponding type or with an error response PDU (ATT_ERROR_RSP) within 30 seconds. Failure to respond within 30 seconds constitutes a time out.

An example of an indication / confirmation PDU pairing is the ATT_HANDLE_VALUE_IND and ATT_HANDLE_VALUE_CFM PDUs shown in Figure 47 - Indications and Confirmations.

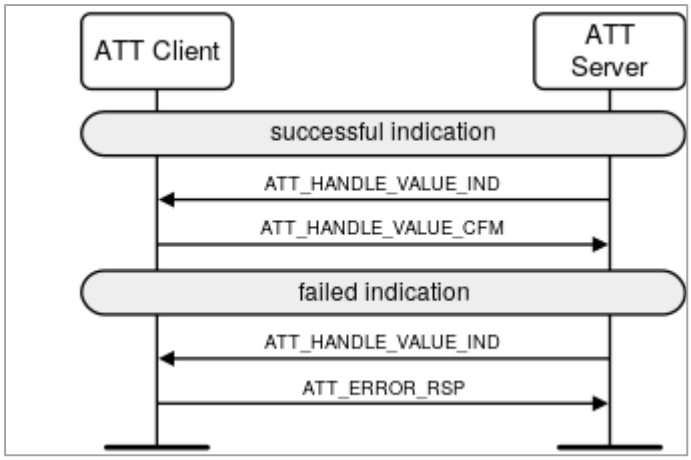


Figure 47 - Indications and Confirmations

11.2.5 PDU Format

All ATT PDUs have the same structure, consisting of an opcode which identifies the PDU type, a set of parameters and an optional authentication signature. Note that the signature field is rarely used and is redundant when the attribute protocol runs over an encrypted link since all encrypted packets at the link layer contain authentication data.

11.2.6 Maximum Transmission Unit

The maximum length of an ATT PDU depends on the Maximum Transmission Unit (MTU) value that has been established. One of two mechanisms may be used to establish the MTU, depending on the *bearer*[8] being used for ATT.

11.3 Transactions

ATT defines the concept of a *transaction*. *Request* PDUs from a client expect a *response* PDU to be returned by the server within 30 seconds. *Indications* sent by a server are expected to be replied to by the client with a *confirmation* PDU within 30 seconds. Each request/response pair or indication/confirmation pair forms a transaction. If a transaction times out then it is regarded as to have failed and no further PDUs of any type may be sent using the current *bearer*.

ATT uses a sequential transaction model. This means that if an ATT transaction has been started, no further ATT PDUs may be processed by the same bearer instance until the current transaction has completed. The transaction is deemed to have completed when the expected response or confirmation PDU has been received from the remote device or when the transaction has timed out after waiting for 30 seconds.

11.4 Bearers

ATT is handled by the L2CAP layer beneath in one of two ways, each known as a *bearer*. The two ATT bearers are the *Unenhanced ATT bearer* and the *Enhanced ATT bearer*. Which bearer is in use has implications for the way in which ATT can be used and in some instances, the reliability of the protocol. The Generic Attribute Profile deals with how ATT is used and states, for example that:

The Unenhanced ATT bearer

[8] See 11.4 Bearers

- Uses a fixed L2CAP channel and there may therefore only be one instance of this bearer.
- Transactions are strictly sequential, irrespective of how many clients at the application layer are using ATT. This can mean that a transaction initiated by one application can delay transactions that another application wishes to start.
- The ATT_EXCHANGE_MTU_REQ and ATT_EXCHANGE_MTU_RSP PDUs may be exchanged to influence the choice of ATT MTU to be used over the Unenhanced ATT bearer.
- Any notifications received by a client which cannot be processed due to issues such as buffer overflow are discarded. As such, the ATT_HANDLE_VALUE_NTF PDU is considered to be unreliable when used over the Unenhanced ATT bearer.
- Support for some PDU types such as ATT_MULTIPLE_HANDLE_VALUE_NTF, ATT_READ_MULTIPLE_VARIABLE_REQ and ATT_READ_MULTIPLE_VARIABLE_RSP is optional when using the Unenhanced ATT bearer.
- The L2CAP channel supporting the Unenhanced ATT bearer may be unencrypted or encrypted.

The Enhanced ATT bearer

- Uses dynamic L2CAP channels and multiple channels and therefore multiple bearer instances are permitted.
- Transactions are handled sequentially but on a per bearer basis. Therefore within a stack, parallel transactions, each handled by a separate Enhanced ATT bearer instance is possible. This has obvious benefits, avoiding the possibility of one application's use of ATT being blocked by another.
- ATT MTU is set to the MTU value which is used at the L2CAP layer automatically and the ATT_EXCHANGE_MTU_REQ and ATT_EXCHANGE_MTU_RSP PDUs are not permitted over an Enhanced ATT bearer.
- An L2CAP flow control method called the Enhanced Credit Based Flow Control Mode is used with the Enhanced ATT bearer. This has the effect that PDUs that are unreliable when used over the Unenhanced ATT bearer can be regarded as reliable when using an Enhanced ATT bearer.
- Support for some PDUs such as ATT_MULTIPLE_HANDLE_VALUE_NTF, ATT_READ_MULTIPLE_VARIABLE_REQ and ATT_READ_MULTIPLE_VARIABLE_RSP is mandatory when using the Enhanced ATT bearer.
- The L2CAP channel supporting the Enhanced ATT bearer must be an encrypted channel.

11.5 Discovering Support for EATT

The defined Generic Attribute Profile service allows a client to determine whether or not a connected server supports EATT and conversely, to allow the client to inform the server that it supports EATT.

A characteristic called *Server Supported Features* must be included in the Generic Attribute Profile service if EATT is supported by the server. Bit 0 of the first octet of the value of this characteristic set to 1 means that EATT is supported. A GATT/ATT client can determine whether or not the server supports EATT by reading this characteristic.

The Client Supported Features characteristic value consists of bits which indicate support or otherwise for certain features. Bit 1 indicates whether or not the Enhanced ATT Bearer is supported

by the client. Bit 2 indicates whether or not the ATT_MULTIPLE_HANDLE_VALUE_NTF PDU is supported. The client must write an appropriate value to this characteristic to inform the server of the features it supports.

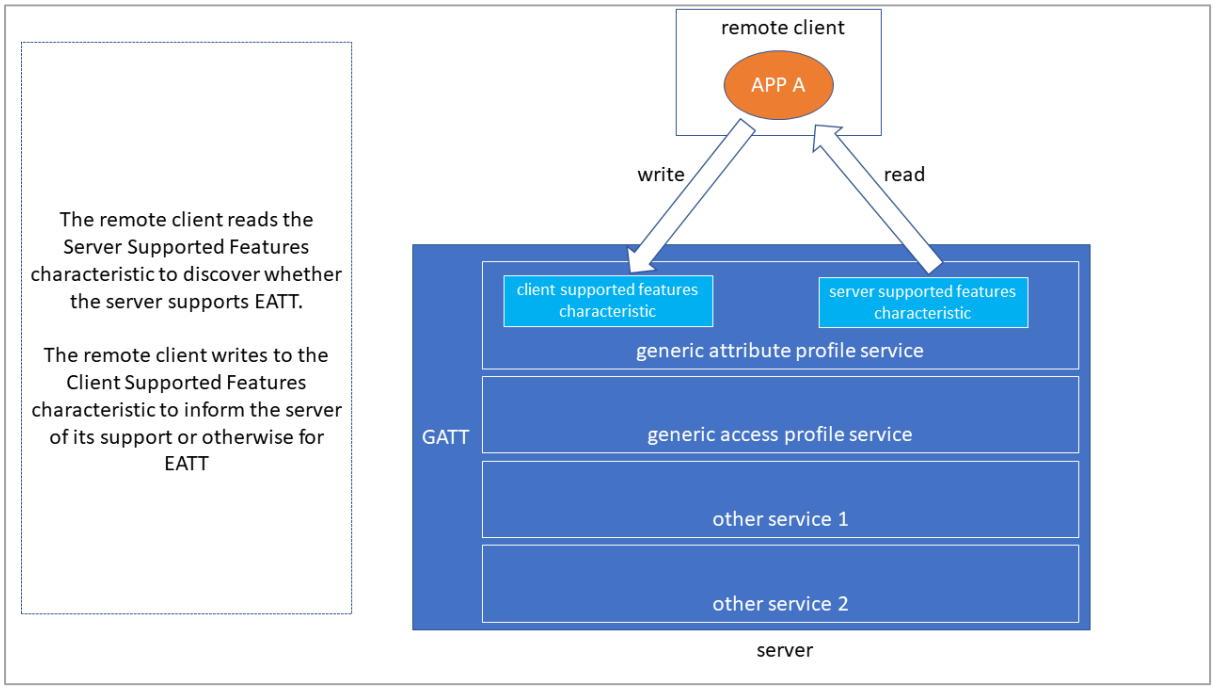


Figure 48 - EATT feature support discovery

12. The Generic Attribute Profile

12.1 Basics

The Generic Attribute Profile (GATT) defines higher level data types based on the *attributes* held in the *attribute table* (see 11. The Attribute Protocol). Those data types are called *services*, *characteristics* and *descriptors*. It also defines a series of procedures involved in using these data types via the Attribute Protocol (ATT). Applications typically use platform APIs that are mapped to these procedures.

Services are grouping mechanisms which provide a context within which to use the characteristics that they contain and have a defined type. Often services correspond to a primary feature or capability of a device.

Characteristics are individual items of state data and have a type, an associated value and a set of properties which indicate how the data may be used in terms of sets of related GATT procedures. For example, it may be defined that a connected peer device can read the value of a particular characteristic but cannot write to it.

Characteristics belong to a service. The same characteristic type can be a member of more than one service and based on the different contexts these services provide, the rules for using the characteristic might vary. A service specification will provide these details.

Descriptors belong to some characteristics and can contain metadata like a textual description for the characteristic or might provide some means of controlling the behaviour of a characteristic. Characteristics have zero or more descriptors attached to them. For example, GATT defines an operation called *characteristic value notification* which involves a device sending an ATT PDU containing a characteristic value to the connected peer asynchronously and without requiring a response from the other device. If a characteristic supports notifications, typically notifications will be transmitted either when the characteristic value changes or periodically, controlled by a timer. But notifications will only be sent if the peer device has requested them, and this is done by setting a flag in a particular type of descriptor called the *Client Characteristic Configuration descriptor* which the characteristic must have if it supports notifications.

The hierarchical structure of service, characteristics and descriptors is shown in Figure 49.

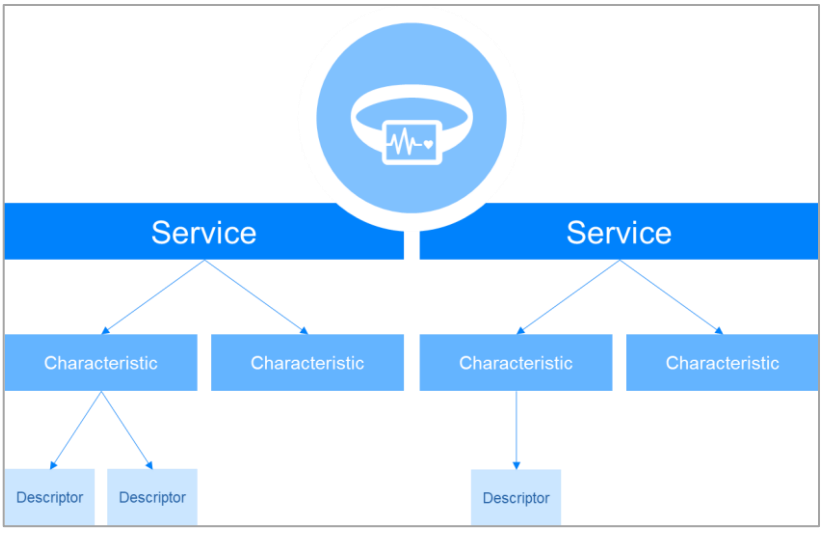


Figure 49 - Services, Characteristics and Descriptors

GATT defines two roles. The GATT client sends ATT commands and requests to the GATT server. The GATT server accepts and processes commands and requests received from a GATT client and sends to the GATT client ATT notifications, indications and responses.

Two special services are mandatory in all GATT servers. These are the *generic access service* and the *generic attribute service*.

12.2 Bluetooth SIG vs Custom

Some services, characteristics and descriptors are defined by the Bluetooth SIG and have 16-bit UUID values that identify their type. The Bluetooth SIG list of each defined type is available from https://www.bluetooth.com/specifications/assigned-numbers. Implementers may purchase 16-bit UUIDs and other types of assigned number as described at https://support.bluetooth.com/hc/en-us/articles/360062030092-Requesting-Assigned-Numbers.

An implementer may define custom services, characteristics and descriptors. Custom services, characteristics and descriptors may either be identified by 128-bit UUID values allocated by the implementer or the implementer may purchase 16-bit UUID values from the Bluetooth SIG. 16-bit UUIDs also have an equivalent 128-bit value in the form 0000**XXXX**-0000-1000-8000-00805F9B34FB where XXXX is the 16-bit UUID value. Implementers may not use UUIDs in this range other than when a UUID has been purchased from the Bluetooth SIG.

A GATT server may contain Bluetooth SIG defined services, characteristics and descriptors (attributes) only or it may contain a mixture of Bluetooth SIG defined attributes and custom attributes.

12.3 Procedures

GATT procedures covering service discovery, characteristic discovery, descriptor discovery, reading and writing characteristic values and notifying and indicating characteristic values are defined, amongst others. The GATT specification provides a clear mapping between its procedures and the underlying ATT protocol which these procedures must use.

12.4 Examples

12.4.1 Bluetooth SIG defined attributes only

Figure 50 shows an example of a set of services with their characteristics. One characteristic has an associated descriptor. Each attribute in this example is defined by the Bluetooth SIG.

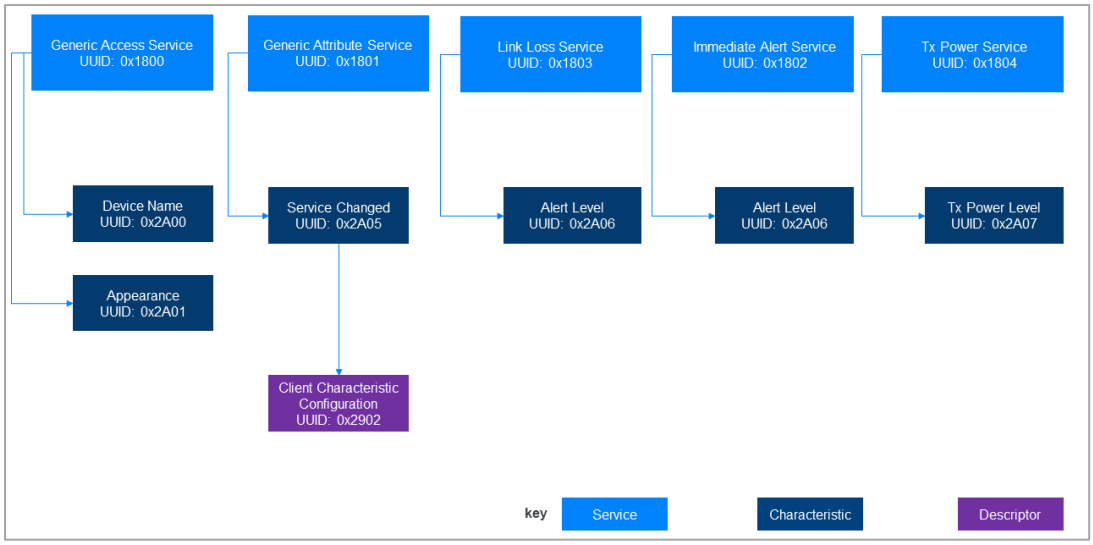


Figure 50 - An example set of services, characteristics and descriptor

The services shown are those which a device that implements the standard proximity profile would probably have (the *immediate alert* and *TX power* services are not mandatory). Notice how the *alert level* characteristic appears twice, once within the *link loss* service and once in the *immediate alert* service. The UUID is the same in each case. This is what identified the characteristic as the *alert level* characteristic. But the service within which the characteristic is grouped provides a distinct context and the rules and behaviour relating to the *alert level* characteristic differ across the two services.

The *service changed* characteristic has an associated *client characteristic configuration* descriptor because the characteristic supports notifications. Any characteristic supporting notifications or indications must have a *client characteristic configuration* descriptor because its value (which clients may write to) controls whether or not notifications or indications are currently enabled or not.

12.4.2 Mixture of Bluetooth SIG and Custom Attributes

Figure 51 shows a GATT server with a mixture of GATT attributes defined by the Bluetooth SIG with a single custom service which contains a single custom characteristic. The custom service is called the proximity monitoring service and has a UUID type identifier value of 0x 3E099910-293F-11E4-93BD-AFD0FE6D1DFD. It's characteristic is called the Client Proximity characteristic and it has a UUID value of 0x 3E099911-293F-11E4-93BD-AFD0FE6D1DFD. Note that this service and characteristic are used in the project work in the educational developer resource *An Introduction to Bluetooth Low Energy Development*. See 16. Additional Resources for further details.

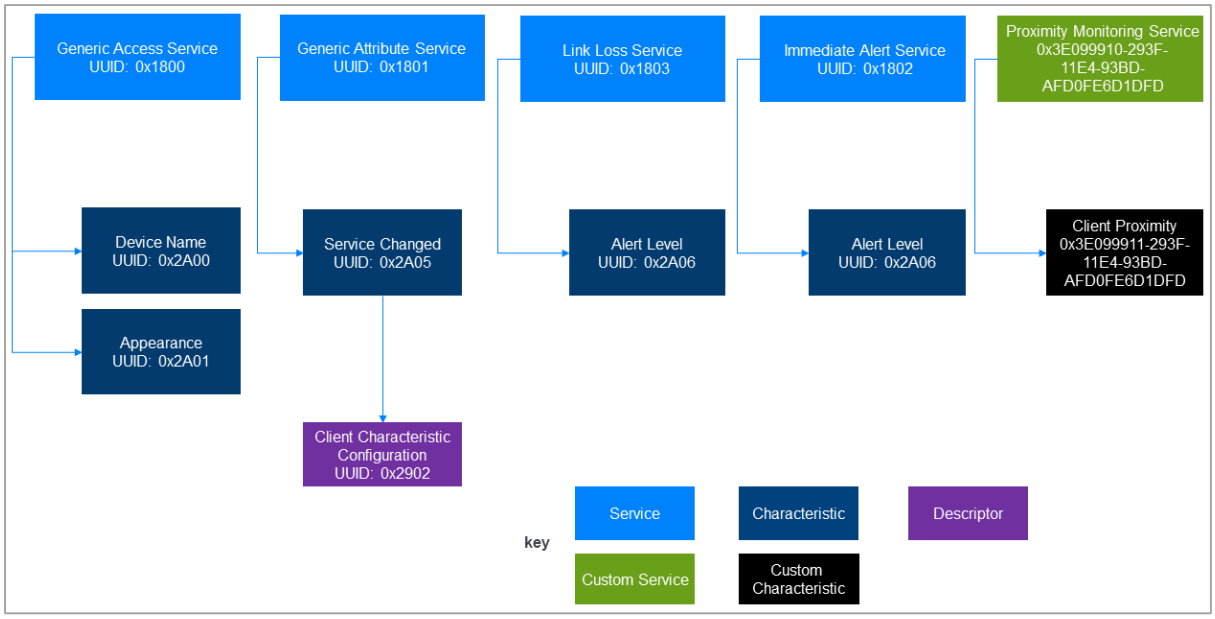


Figure 51 - A mixture of Bluetooth SIG defined and custom attributes

13. The Generic Access Profile

13.1 Basics

The Generic Access Profile (GAP) section of the Bluetooth Core Specification defines procedures concerned with device discovery and establishing connections between two devices. How to perform connectionless communication of data in general, how to use periodic advertising (see 7.7.3 PADVB - LE Periodic Advertising Broadcast) and how to set up isochronous communication (see 7.7.5 LE BIS and LE CIS - Isochronous Communication) are also topics covered by GAP.

In addition, some key user interface standards and certain aspects of Bluetooth LE security are also covered by this section of the core specification.

The transmission of advertising packets (*advertising*) and their receipt through *scanning* is at the heart of how much of GAP works. There are a number of different advertising and scanning packet types and these are defined by the link layer. It should be noted that the payload field is called AdvData and that this field is not present in all PDU types. When it is present, data that it contains is encoded as a series of one or more length/tag/value constructs known as AD Types. AD Types are defined in the Core Specification Supplement (CSS) document, available from the specifications page at bluetooth.com.

GAP has relevance to both Bluetooth LE and Bluetooth BR/EDR. In the remainder of this section only GAP as it applies to Bluetooth LE is covered. Furthermore it should be noted that whilst activities like advertising and scanning are of central relevance to GAP, these procedures are actually performed by the link layer as are the PDU types involved.

13.2 Roles

GAP defines four device roles. These are listed and explained in Table 7.

Role	Description
Broadcaster	A device which uses some form of advertising to transmit data in a connectionless manner. This includes legacy advertising, extended advertising and periodic advertising. A broadcaster may also transmit a broadcast isochronous stream. A broadcaster has a transmitter but possessing a receiver is optional. A broadcaster does not accept connections from Central devices (unless it is also acting in the Peripheral role).
Observer	An Observer receives advertising packets or broadcast isochronous stream data packets. It does not connect to other devices, contains a transmitter and may or may not contain a receiver. The Observer is able to receive broadcast data in a connectionless manner.
Peripheral	A Peripheral can be connected to by a Central device. It contains a transmitter and a receiver.
Central	A Central is able to initiate the establishment of a connection with a Peripheral device. It contains both a transmitter and a receiver.

Table 7 - GAP roles

Note that the role names *Central* and *Peripheral* are also used by the link layer. The terms in each of these two different contexts do not mean the same thing.

13.3 Discovery

A Broadcaster or Peripheral device is either in the non-discoverable mode or it is in one of the two discoverable modes that GAP defines. When advertising in non-discoverable mode, transmitted packets are visible over the air (this is not a security feature) but scanning devices performing either the General Discoverable procedure or the Limited Discoverable procedure will ignore these packets.

A discoverable device may either be in the *general discoverable* mode or the *limited discoverable* mode. When in the general discoverable mode, a device is discoverable for an indefinite amount of time whereas in the limited discoverable mode it is discoverable for a maximum of three minutes.

Discovering devices are able to recognize which of the discoverable modes an advertising device is in by examining an AD Type in the AdvData field called *Flags*. Limited discoverable mode is often used to give priority to devices the user has recently interacted with, perhaps pressing a button or simply picking up and moving the device.

When a Central or Observer devices attempts to discover other devices, it may use either *passive scanning* or *active scanning*. Which of the two are permitted depends on whether the device is attempting to discover devices in general discoverable mode or limited discoverable mode.

Passive scanning involves receiving advertising PDUs without sending any scanning PDUs. Active scanning involves receiving advertising PDUs and requesting more information in response by sending scanning PDUs. The various PDU types are defined by the link layer and are summarized in 7.7.2 ADVB - LE Advertising Broadcast.

The Bluetooth Core Specification notes that some devices only use legacy advertising whereas others might use extended advertising or interleave both forms of advertising. It is recommended that devices performing one of the discovery procedures interleave scanning for both types of advertising. It is also recommended that a device scan on all PHYs that it supports.

Figure 52 shows GAP discovery modes in conjunction with other relevant variables.

13.4 Connection Modes

An advertising device can indicate whether or not it is available to be connected to either by the PDU used (legacy advertising) or the value of the AdvMode field (extended advertising).

Figure 52 shows GAP connection modes in conjunction with other relevant variables.

A device may request a connection with another device by performing one of the connection-related procedures defined by GAP. This will generally involve sending either a CONNECT_IND PDU (legacy advertising) or a AUX_CONNEXT_REQ PDU (extended advertising) as a response to a received PDU of one of several types that permit this. The link layer defines advertising and connection request PDU types and the rules regarding the PDU types for which a connection request may be sent in response. See 7.7.2.2.3 Legacy Advertising and Associated PDU Types and 7.7.2.3.5 Extended Advertising and Associated PDU Types.

13.5 Directed vs Undirected

Advertising as used by GAP may either be *undirected*, meaning that PDUs are applicable to any Observer or Central device that receives them or *directed* meaning that only a specific device should process such PDUs. PDUs involved in directed advertising include the TargetA field with the Bluetooth address of the intended recipient device. In undirected advertising, the TargetA field is absent.

Figure 52 shows directed and undirected advertising in the context of GAP discovery and connection modes.

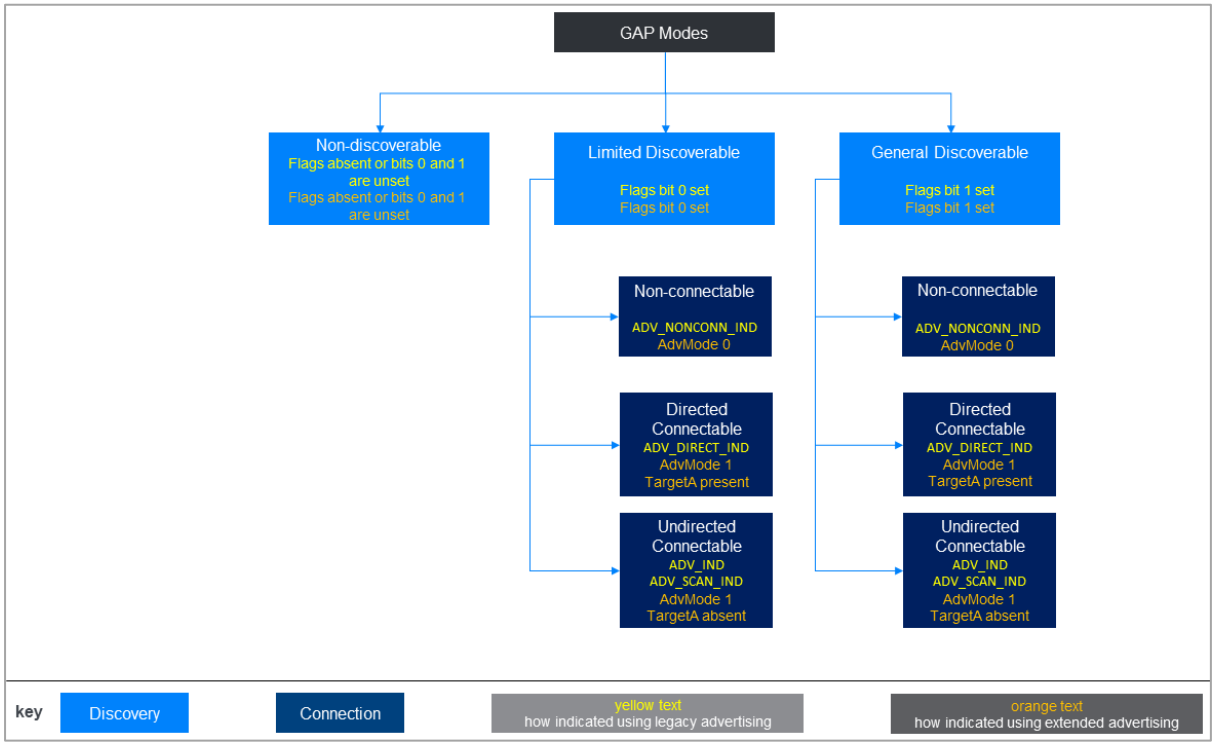


Figure 52 - GAP discovery and connection modes

Note that the AD Type *Flags* appears in legacy advertising packets received in the primary advertising channel. When extended advertising is in use however, the AdvData field is not present in ADV_EXT_IND PDUs received on the primary channels. When Flags is in use with extended advertising, it appears in auxiliary AUX_EXT_IND PDUs on the general-purpose channels.

13.6 Scannable vs Non-scannable

Certain advertising PDU types are said to be *scannable*. This means that a device receiving such a PDU is permitted to respond with a scan request PDU of an appropriate type, to request more advertising data. Advertising PDUs are defined by the link layer. See 7.7.2.2.3 Legacy Advertising and Associated PDU Types and 7.7.2.3.5 Extended Advertising and Associated PDU Types for details.

13.7 GAP and LE Security

The GAP specification defines a number of security terms, modes and procedures. Generally, GAP security procedures involve other layers of the stack such as the Security Manager Protocol (SMP) and the link layer but the high level procedure for using those layers to accomplish certain results is defined in Volume 3 Part C (Generic Access Profile) of the Bluetooth Core Specification, which should be consulted for details.

13.8 Periodic Advertising

Periodic Advertising is performed by the link layer (see 7.7.3 PADVB - LE Periodic Advertising Broadcast) but GAP specifies the procedure for a Broadcaster to enter periodic advertising mode and for an Observer to synchronize with a periodic advertising train. In addition, the Periodic Advertising Synchronization Transfer (PAST) procedure which allows an Observer to acquire periodic advertising

synchronization parameters from a Broadcaster and pass them over an ACL connection to another device is defined in GAP.

13.9 Isochronous Broadcast

Isochronous communication using broadcast isochronous streams and connected isochronous streams is performed by the link layer (see 7.7.5 LE BIS and LE CIS - Isochronous Communication) but GAP specifies the procedures which a Broadcaster and Observers must follow to engage in this form of communication.

14. The Security Manager Protocol

14.1 Basics

The security manager protocol (SMP) is part of the stack's security manager component. It supports the execution of security related procedures such as pairing, bonding and key distribution.

The security manager component provides a cryptographic toolbox for security functions which other layers can use and defines pairing algorithms.

Section *15. Security in Bluetooth LE* has more to say about security in general.

14.2 Example

Figure 53 shows SMP being used during the pairing of two devices. Note the exchange of input/output capabilities and other flags during the SMP Pairing Feature Exchange. This is an important step which determines which pairing algorithm is selected and how steps like authentication and incorporated in the procedure.

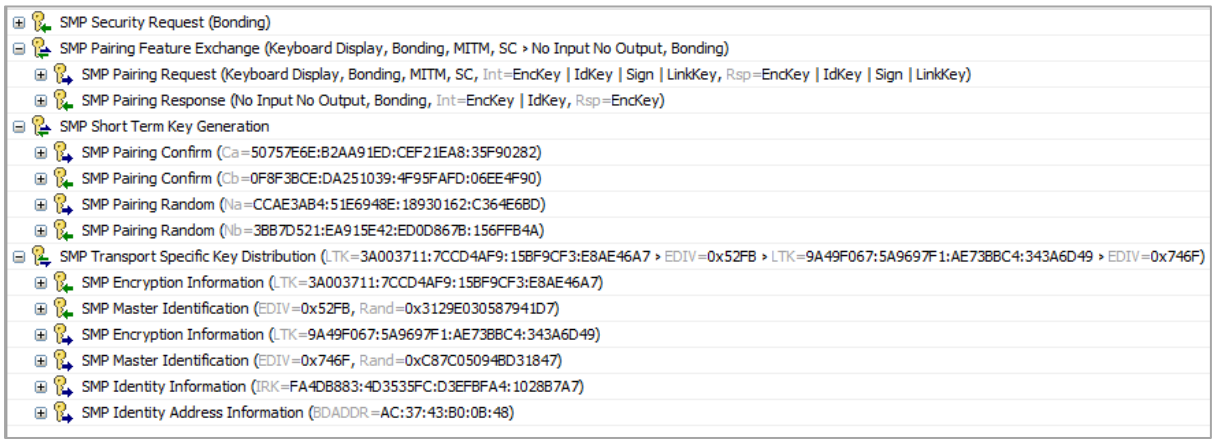


Figure 53 - SMP in use during pairing

15. Security in Bluetooth LE

Security is a critical topic and one which needs to be carefully considered and understood.

Bluetooth LE provides a collection of security capabilities and features, most of which are optional. You should think of this as a *toolbox* containing security tools with which to address specific security issues and meet specific security requirements. It is the responsibility of the product team, having ascertained the security requirements for a product, to meet those requirements. Where appropriate, this should be achieved through the use of selected Bluetooth LE security features.

Given the importance of this subject, an educational resource dedicated to this topic has been created by the Bluetooth SIG and its content will not be repeated here. Consult section *16. Additional Resources* for more information.

16. Additional Resources

This section lists other resources which support learning about Bluetooth LE from various perspectives.

Resource	Description	Location
Bluetooth Core Specification	The key technical specification. Defines all layers of the Bluetooth stack and associated protocols and procedures. Covers both Bluetooth LE and Bluetooth BR/EDR.	https://www.bluetooth.com/specifications/specs/
Profile and service specifications	A service specification defines a single GATT service along with the characteristics and descriptors that it contains. The behaviours to be exhibited by the GATT server device hosting the service in response to various conditions and state data values are defined in the service specification. Profile specifications define the roles that related devices assume and in particular, define the behaviour of the client device and the data on the connected server that it should work with.	https://www.bluetooth.com/specifications/specs/
LC3	Defines the Low Complexity Communication Codec used by LE Audio.	https://www.bluetooth.com/specifications/specs/
Study Guide - An Introduction to Bluetooth Low Energy Development	An educational resource for developers wishing to learn about software development for connection-oriented scenarios involving smartphones and peripheral devices. Includes a series of hands-on projects with solutions.	https://www.bluetooth.com/bluetooth-resources/bluetooth-le-developer-starter-kit/
Study Guide - An Introduction to Bluetooth Mesh Software Development	An educational resource for developers wishing to learn about Bluetooth mesh and about the implementation of mesh models in microcontrollers. Includes a series of hands-on projects with solutions.	https://www.bluetooth.com/bluetooth-resources/bluetooth-mesh-developer-study-guide/
Study Guide - An Introduction to the Bluetooth Mesh Proxy Function	An educational resource for developers wishing to learn how to create GUI applications for devices such as smartphones which allow interaction with nodes in a Bluetooth mesh network. Includes a series of hands-on projects with solutions.	https://www.bluetooth.com/bluetooth-resources/bluetooth-mesh-proxy-kit/
Paper - Bluetooth Mesh Networking - An Introduction for Developers	An educational resource for anyone interested in learning about the key concepts and capabilities of Bluetooth Mesh.	https://www.bluetooth.com/bluetooth-resources/bluetooth-mesh-networking-an-introduction-for-developers/

Paper - Bluetooth Mesh Models - A Technical Overview	An educational resource for anyone interested in better understanding the standard models available for use in Bluetooth mesh products.	https://www.bluetooth.com/bluetooth-resources/bluetooth-mesh-models/
Study Guide - Understanding Bluetooth LE Security	An educational resource which explains and illustrates all aspects of security in Bluetooth LE (excluding Bluetooth mesh). Suitable for both complete beginners to the subject of security and those with prior experience. Includes a series of hands-on projects with solutions.	https://www.bluetooth.com/bluetooth-resources/le-security-study-guide/
Paper - Bluetooth Security and Privacy Best Practices Guide	A guide which is intended to help implementers better understand why certain available security and privacy choices are better or worse than others for specific applications, and what risks and pitfalls remain in the specifications.	https://www.bluetooth.com/bluetooth-resources/bluetooth-security-and-privacy-best-practices-guide/
Study Guide - Bluetooth Technology for Linux Developers	An educational resource for Linux developers wishing to learn about developing software which uses the Linux Bluetooth stack, BlueZ. Includes a series of hands-on projects with solutions.	https://www.bluetooth.com/bluetooth-resources/bluetooth-for-linux/
Study Guide - Designing and Developing Bluetooth Internet Gateways	An educational resource which explains Bluetooth internet gateways which are used to provide access to Bluetooth LE and Bluetooth mesh devices from the internet. Illustrates possible architectures and implementation approaches. Includes a series of hands-on projects with solutions.	https://www.bluetooth.com/bluetooth-resources/bluetooth-internet-gateways/
Study Guide - An Introduction to Web Bluetooth	An educational resource for developers wishing to learn about developing web applications which use the JavaScript Web Bluetooth APIs. Includes a series of hands-on projects with solutions.	https://www.bluetooth.com/bluetooth-resources/web-bluetooth-tutorial/
Study Guide - An Introduction to Bluetooth Beacons	An educational resource for developers wishing to learn about Bluetooth beacons. Includes a series of hands-on projects with solutions.	https://www.bluetooth.com/bluetooth-resources/beacon-smart-starter-kit/
Paper - Bluetooth Core Specification Version 5.0 Feature Enhancements	Explains the new features and other changes released in version 5.0 of the Bluetooth Core Specification. Includes the LE 2M PHY, LE Coded PHY and extended advertising.	https://www.bluetooth.com/bluetooth-resources/bluetooth-5-go-faster-go-further/
Paper - Bluetooth Core Specification Version 5.1 Feature Overview	Explains the new features and other changes released in version 5.1 of the Bluetooth Core Specification. Includes AoA and AoD direction finding.	https://www.bluetooth.com/bluetooth-resources/bluetooth-core-specification-v5-1-feature-overview/
Paper - Bluetooth Core Specification	Explains the new features and other changes released in version 5.2 of the Bluetooth Core Specification. Includes the	https://www.bluetooth.com/bluetooth-resources/bluetooth-

Version 5.2 Feature Overview	Enhanced Attribute Protocol, LE Power Control and LE Isochronous Channels.	core-specification-version-5-2-feature-overview/
Paper - Bluetooth Core Specification Version 5.3 Feature Overview	Explains the new features and other changes released in version 5.3 of the Bluetooth Core Specification. Includes LE Enhanced Connection Update using subrating and LE Channel Classification.	https://www.bluetooth.com/bluetooth-resources/bluetooth-core-specification-version-5-3-feature-enhancements/
Paper - Bluetooth mesh Models - A Technical Overview	Provides a guide to understanding the standard set of Bluetooth mesh models.	https://www.bluetooth.com/bluetooth-resources/bluetooth-mesh-models/
eBook - Introducing Bluetooth LE Audio	A book written by Nick Hunn which provides a clear introduction to the features and technicalities of Bluetooth LE Audio.	https://www.bluetooth.com/bluetooth-resources/le-audio-book/