Bluetooth® Core Specification Version 5.3 Feature Enhancements

Bluetooth Core Specification version 5.3 includes several updates. This technical overview paper summarizes and explains each change. The paper is marketing document and is not intended to replace or overrule the Bluetooth Core specification in any way.

Author: Martin Woolley

Version: 1.0.1

Revision Date: 29 August 2023

Bluetooth®

Revision History

Version	Date	Author	Changes
1.0.0	24 June 2021	Martin Woolley	Initial Version
1.0.1	29 August 2023	Damon Barnes, Bluetooth SIG	Minor copy updates

Table of Contents

Table of Contents

Table of Contents

At a Glance

AdvDataInfo in Periodic Advertising

The AdvDataInfo (ADI) field of the common extended advertising payload format may now be included in AUX_SYNC_IND protocol data units (PDUs) which are broadcast when a device is performing periodic advertising. On receiving AUX_SYNC_IND PDUs Bluetooth Low Energy (LE) controllers may use the information in the ADI field to recognise packets that contain retransmitted copies of identical or semantically equivalent data which has already been successfully received. The controller may then discard the packet rather than pass its content to the host. It may also afford the opportunity for the controller to switch to scanning on a primary channel earlier than would otherwise have been the case and therefore improve RX duty cycle as it applies to those channels.

Encryption Key Size Control Enhancements

In Bluetooth BR/EDR, encryption key sizes are negotiated by the controllers in connected devices. This change allows a host to inform its Bluetooth BR/EDR controller of the minimum acceptable key size using the Host Controller Interface (HCI). This enhancement also improves the efficiency with which Bluetooth BR/EDR controllers can inform the host of the outcome of key length negotiations.

LE Enhanced Connection Update

Some product types spend much of their time in a low duty cycle connection so that power is conserved. But when a higher bandwidth is needed to support a particular application use case, the connection parameters must be changed as quickly as possible. The Bluetooth LE enhanced connection update improvement introduces the concept of connection subrating which allows connection parameter updates to be made with minimal delay, delivering a better user experience but at the same time, retaining the power saving properties of low duty cycle connections.

LE Channel Classification

Peripheral devices are now able to provide a connected Central device with radio channel classification data which may be used by the Central device when performing channel selection during adaptive frequency hopping. This improves throughput and reliability by reducing susceptibility to interference taking place at the Peripheral when the Peripheral and Central devices are not physically close to each other.

Removal of the Alternate MAC and PHY (AMP) Extension

The Alternate Media Access Control and Physical layer extension usually abbreviated as AMP allows a Bluetooth system to include one or more secondary controllers alongside a primary Bluetooth BR/EDR controller. Secondary controllers allow the use of the IEEE 802.11 MAC and PHY protocols instead of the standard Bluetooth layers that are part of the primary controller.

The frequency with which AMP has been used in qualified Bluetooth products has not been high and the Bluetooth SIG have taken the decision to remove this feature from the Bluetooth core specification version 5.3. Members will still be able to qualify products that use AMP against an earlier Bluetooth core specification version such as 5.2, until further notice.

1.0 AdvDataInfo in Periodic Advertising

1.1 Background

1.1.1 Extended Advertising

Bluetooth Low Energy (LE) has a capability known as *extended advertising* which uses its 37 general purpose channels in the ISM band for broadcast communication as well as the three primary advertising channels. Using the general data channels in this way reduces the probability of packet collisions occurring.

Extended advertising may be used in a number of ways, one of which is called *periodic advertising*.

Four distinct PDU types are defined for use in extended advertising and these are named ADV_EXT_IND, AUX_ADV_IND, AUX_SYNC_IND and AUX_CHAIN_IND. Each of these PDU types uses the same payload format, known as the *Common Extended Advertising Payload Format*.

The *Common Extended Advertising Payload Format* defines a number of fields, the most important of which are listed in Table 1.

Field	Description
AdvA	The advertiser's device address.
TargetA	Address of the device to which the advertisement is directed.
CTEInfo	If present, indicates that the packet includes a Constant Tone Extension.
AdvDataInfo (ADI)	IIdentifies the advertising set to which the advertisement belongs and allows retransmissions of equivalent data within that set to be recognised. ADI consists of two sub-fields named Advertising Data ID (DID) and Advertising Set ID (SID). Identical SID and DID values identify identical or equivalent advertising data within a particular advertising set.
AuxPtr	Indicates that some or all of the advertisement data is to be expected in a subsequent auxiliary packet and provides data such as a channel index which allows that packet to be received.
SyncInfo	Indicates the presence of a *periodic advertising train* involving AUX_SYNC_IND advertising PDUs.

Field	Description
Tx Power	The radiated power level of the packet measured at the transmitter.
ACAD	Additional Controller Advertising Data - may contain additional controller-provided data fields as defined in the Bluetooth Core Specification Supplement (CSS) such as a Service UUID and Service Data
AdvData	Contains advertising data provided by the host.

Table 1 - The Common Extended Advertising Format Fields

Which of the fields from the Common Extended Advertising Format are included in a packet depends on the PDU type. For each of the extended advertising PDU types, the Bluetooth core specification designates each field to be mandatory (M), optional (O), conditional (Cn) or reserved for future use (X).

1.1.2 Periodic Advertising

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 1 reproduces Figure 4.5 from Volume 6 Part B of the Bluetooth core specification and illustrates the effect of the advDelay parameter.

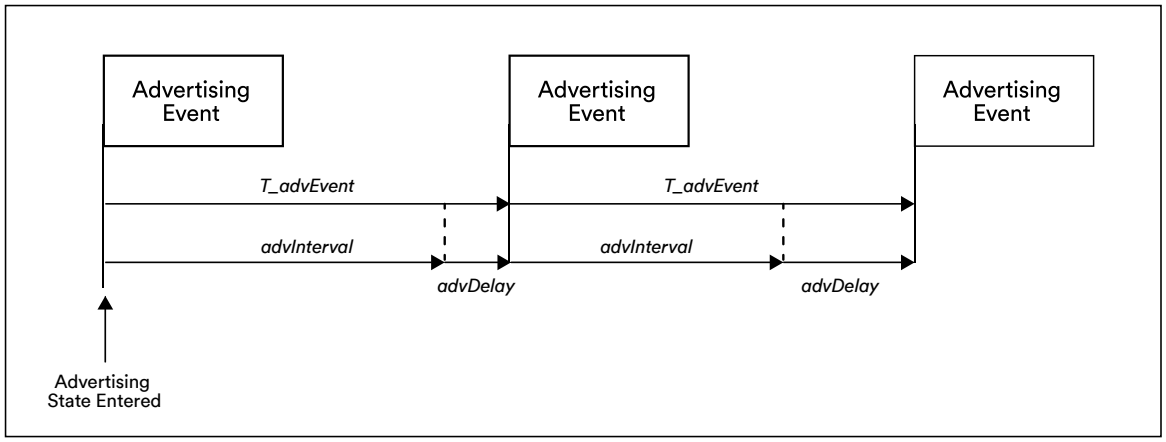

Figure 1 - Advertising events perturbed in time using advDelay

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. Consequently, a special extended advertising mode called *Periodic Advertising* is defined. Periodic advertising does not use the random advDelay value in its scheduling algorithm

and periodic advertising events occur at completely deterministic timing intervals. Information about the periodic advertising schedule is included in AUX_ADV_IND PDUs in a field called SyncInfo. This allows receivers to determine the periodic advertising schedule of the broadcasting device and to synchronise their scanning activities precisely with it.

Periodic advertising involves three of the extended advertising PDU types and all 40 radio channels, as depicted in Figure 2.

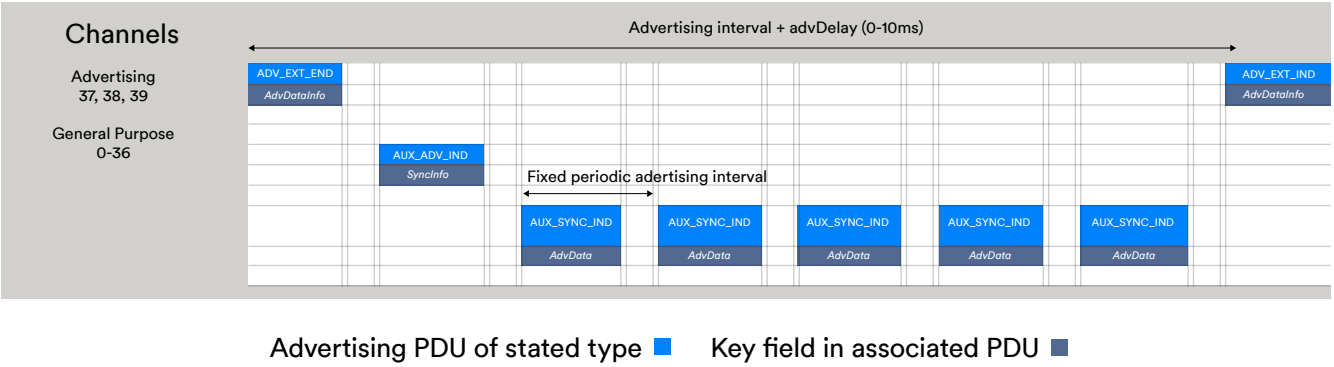

Figure 2 - Periodic Advertising PDUs and Channel Use

The AUX_SYNC_IND advertising PDU type is transmitted at fixed intervals and its use of the *Common Extended Advertising Format* as it was previously defined in version 5.2 of the Bluetooth core specification is reproduced in Figure 3. Note that the AdvDataInfo (ADI) field shall not be included in AUX_SYNC_IND PDUs at any version of the Bluetooth core specification up to and including version 5.2.

		Common Extended Advertising Payload Format Fields								
ADV Mode	**Event Type**	**AdvA**	**TargetA**	**CTE Info**	**ADI**	**Aux Ptr**	**Sync Info**	**Tx Power**	**ACAD**	**ADV Data**
0b00	Non-connectable and Non-Scannable Undirected or Directed	X	X	O	X	O	X	O	O	O
0b01 to 0b11	RFU									

Figure 3 - AUX_SYNC_IND PDUs and the Common Extended Advertising Format

1.1.3 Broadcast Retransmissions

Broadcast communication is usually unidirectional, with no form of acknowledgement protocol which allows receivers to indicate to the transmitter that a packet was successfully received. This makes broadcast communication inherently unreliable.

Consequently, some device types that use extended advertising increase the probability of their broadcast packets being received by retransmitting identical or semantically equivalent copies of the same data multiple times in succession. An LE Audio Broadcast Isochronous Stream (BIS) may use this technique for example.

1.1.4 Energy Efficiency

In a typical Bluetooth system, the host and controller parts of the stack are implemented in discreet components, connected by one of the supported HCI transports (e.g. UART). The host spends much of its time idle, waking up to perform processing only when it needs to.

According to version 5.2 of the Bluetooth core specification, whenever the controller receives a AUX_SYNC_IND advertising packet, it shall notify the host and pass it the advertising packet data using an HCI LE Periodic Advertising Report event (assuming HCI is in use). The controller has no way of recognising periodic advertising packets which contain retransmissions of identical or equivalent data because the ADI field cannot be included in AUX_SYNC_IND PDUs. Consequently, the host may be passed data which it had already successfully received in previous packets, unnecessarily waking it to perform processing of this redundant data. This is sub-optimal and causes the host to waste energy.

1.1.5 The Primary Advertising Channels and RX Duty Cycle

Advertising takes place on one or more of the primary advertising channels, numbered 37, 38 and 39. Typically all three of the primary channels are used, with a copy of the same packet transmitted on each channel, one at a time in quick succession. Extended advertising involves both the primary advertising channels (for ADV_EXT_IND PDUs) and the remaining 37 secondary channels for other PDU types.

The proportion of time that a receiver device spends scanning for packets is called the RX Duty Cycle (*RX* is a common abbreviation for *receive*). Scanning can typically be performed on only one channel at a time though and so the RX duty cycle as it relates to a single channel will be a fraction of the overall RX duty cycle. For example, if the RX duty cycle is 60%, only the three primary advertising channels are in use and for an equal amount of time each, then the RX duty cycle for each of these channels will be approaching 20%.

Advertising packets transmitted on the primary channels are often critical, whether they are the only advertising packets broadcast by a device or are related to other packets transmitted on the secondary channels, as will be the case when extended advertising is in use.

Time spent by the controller receiving a periodic advertising packet which contains the retransmission of data which has already been received, is time not spent scanning on other channels, including the primary channels. In other words, receiving a packet whose content has already been received in a previous advertising packet not only has an energy cost, as described in *1.1.4 Energy Efficiency*, it can also reduce the RX duty cycle of one of the other channels.

1.2 About AdvDataInfo in Periodic Advertising

This section reviews the new *AdvDataInfo* in *Periodic Advertising* change which has been made in the Bluetooth Core Specification version 5.3.

1.2.1 Capabilities and Benefits

The ADI field of the *Common Extended Advertising Payload Format* may now optionally be included in AUX_SYNC_IND periodic advertising PDUs by the controller in a broadcasting device at the request of the host.

In a receiving device, the host may indicate to the controller that it should filter out packets that it identifies as containing data which has already been received in an earlier packet.

This change improves the energy efficiency of host components by reducing the number of packets received from the controller and the associated processing requirement in those cases where retransmissions of identical or semantically equivalent data is being performed by the broadcasting device.

Controller implementations may take advantage of this change by quickly abandoning receipt of a periodic advertising packet which is identified by the controller as a retransmission of a previously received packet. It may do so as soon as the ADI field has been received (as opposed to the whole packet) and immediately switch to scan on another channel. This may improve RX duty cycle on that channel.

1.2.2 Technical Highlights

1.2.2.1 Link Layer

AUX_SYNC_IND PDUs may now include the ADI field. The inclusion of the ADI field in this PDU type is optional however and the controller's link layer implementation will only include ADI in this packet type if the host requests that this be done by sending an *HCI LE Set Periodic Advertising Enable* command to the controller.

The link layer defines a bit mask called *FeatureSet*. Individual bits within this field indicate support or otherwise for a particular link layer feature. A bit has been assigned to indicate whether or not the controller supports the inclusion of the ADI field in periodic advertising packets. This information is used in the validation of HCI commands that relate to this feature. The host may determine whether or not the local controller supports the *Periodic Advertising ADI support* feature by sending the *LE Read Local Supported Features* HCI command to the host. This command returns the complete FeatureSet bit mask from the controller.

1.2.2.2 Host Controller Interface (HCI)

A number of existing HCI commands have been modified so that it is possible for the host to communicate requirements relating to the ADI field in periodic advertising packets to the controller. Table 2 summarises these HCI changes.

HCI Command	Device	Description
LE Set Periodic Advertising Data command	advertiser	Allows the host to set the data included in periodic advertising PDUs by the controller. A new operation field value of 0×04 means “send the same advertising data but update the Advertising Data ID (DID) so it doesn’t look like a duplicate”.
LE Set Periodic Advertising Enable command	advertiser	Allows the host to request that the controller enables or disables periodic advertising for the advertising set specified. Bit 1 of the *Enable* parameter block now indicates whether or not the ADI field should be included in AUX_SYNC_IND PDUs.
LE Periodic Advertising Create Sync command	scanner	Allows the host to request that the controller synchronizes with a periodic advertising train from an advertiser and begin receiving periodic advertising packets. The Options parameter now uses bit 2 to allow the host to indicate whether or not the controller should filter duplicates.
LE Set Periodic Advertising Receive Enable command	scanner	Allows the host to instruct the controller to enable or disable reports for the periodic advertising train identified by the Sync_Handle parameter. The Enable parameter now also allows duplicate filtering to be enabled or disabled at the same time.
LE Set Periodic Advertising Sync Transfer Parameters command	scanner	Allows the host to specify how the controller will process periodic advertising synchronization information received over a particular connection. New mode values allow periodic advertising reports to be enabled with or without duplicate filtering enabled.
LE Set Default Periodic Advertising Sync Transfer Parameters command	scanner	Allows the host to specify the initial value for the mode, skip, timeout, and Constant Tone Extension type. The Mode parameter has been modified so that Mode can both specify whether periodic advertising reports are initially enabled or disabled and whether or not duplicates are to be filtered.

Table 2 - HCI Commands and the ADI in Periodic Advertising Feature

2.0 Host to Controller Encryption Key Control Enhancements

2.1 Background

2.1.1 Encryption Key Lengths

Encryption encodes information so that an unauthorized third party coming into possession of the encoded information cannot decode it and access the original information. Encryption addresses the need for *confidentiality* by ensuring sensitive information can be transmitted in the presence of eavesdroppers without them being able to access it while allowing the intended recipient of the data to decode it.

Unencrypted data provided as an input to a cryptography algorithm is called *plaintext*, and the encrypted output is called *ciphertext*. A key is also provided as an input to an encryption algorithm.

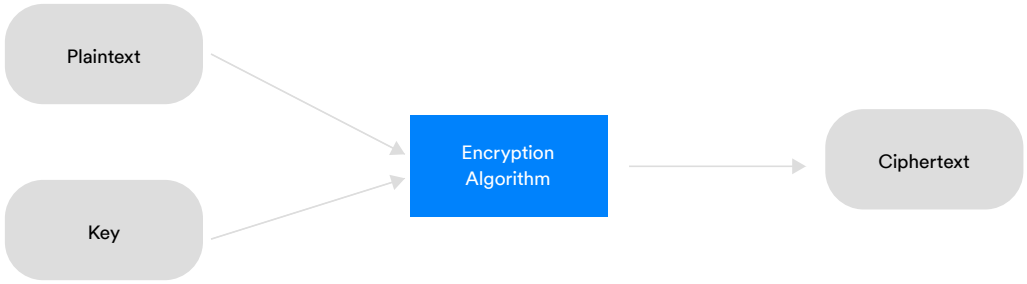

Figure 4 - Encryption

The strength of the protection provided by encryption depends largely on the algorithm used and the amount of entropy or randomness in the key. The amount of entropy in the key is a consequence of the degree to which the values it can have are truly random and the key length, usually expressed as a bit length. So, if for example, you have a 128-bit number generated by a process for which any of the possible 2^{128} values is equally likely to be produced, then your 128-bit number is truly random, and you can say that you have 128 bits of entropy. If, however, you know that the first 64 bits are always all zero, then you would say that your 128-bit number only has 64 bits of entropy.

The Bluetooth technologies use encryption in a number of places. Both Bluetooth Low Energy (LE) and Bluetooth BR/EDR are able to establish an encrypted link between two devices, for example.

2.1.2 Encryption Key Size Negotiation in Bluetooth BR/EDR

The *maximum* key length supported can vary from device type to device type and so this value is generally set in the factory. Individual applications may have different requirements regarding the *minimum* acceptable key length that can be used however and this is usually stipulated in a profile specification. In addition, the core specification stipulates a generally applicable minimum key length of 56 bits.

When establishing an encrypted link between two devices, the two devices will negotiate a suitable key length which each can support and which is deemed

acceptable for the application associated with the connection. This involves the Central device sending a suggested key size to the Peripheral device. The suggested key size is always set to the maximum key size supported by the Central device to begin with. If the Peripheral device can accommodate the Central device's suggested key size, it accepts it. If not, it replies with a suggestion of its own. This exchange of suggestions between Central and Peripheral continues until an agreement is reached or it is concluded that no mutually acceptable key length can be established, in which case the encryption setup is abandoned.

2.1.3 HCI Commands and Events Relating to Encryption in Bluetooth BR/EDR

The Host Controller Interface (HCI) includes a number of commands and events that are relevant to the topic of key size negotiation in Bluetooth BR/EDR. Of particular significance to the Host to Controller Encryption Key Control Enhancements delivered in version 5.3 of the Bluetooth core specification are the following:

Command / Event	Comment
HCI Read Encryption Key Size command	Allows the host to ask the controller what key size was negotiated. All Bluetooth BR/EDR controllers must implement this command.
Encryption Change event	This event is sent by the controller to the host when the encryption state (enabled|disabled) of a connection changes.

2.1.4 Issues with Key Size Negotiation in Bluetooth BR/EDR

Up to and including version 5.2 of the Bluetooth core specification, there was no way for the host to set a minimum key size to be used by the controller when negotiating the key size for a new connection.

Furthermore, a host could only determine the key size agreed by the controller by sending an HCI *Read Encryption Key Size* command after receiving an *Encryption Change* event from the controller. This is illustrated in Figure 5.

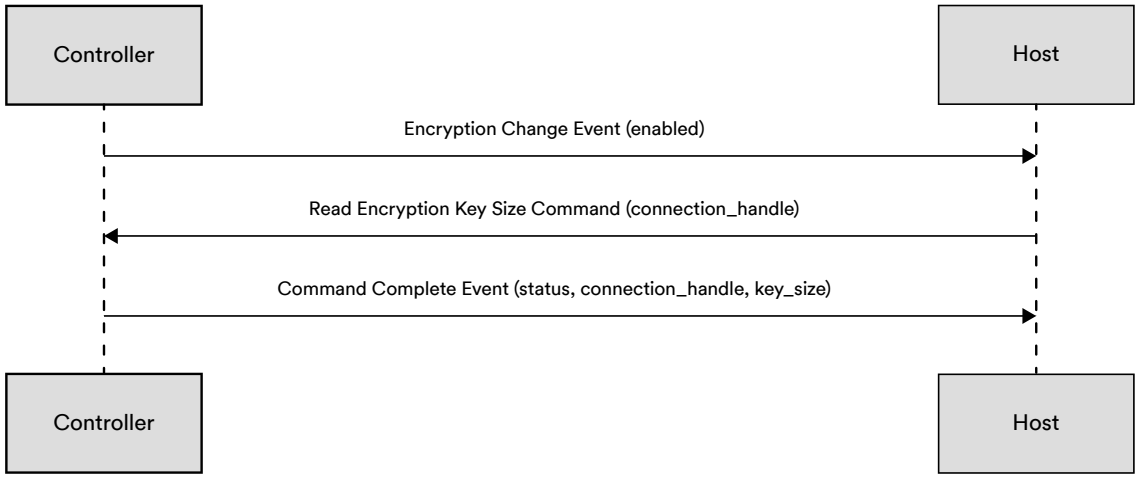

Figure 5 - Host establishing the agreed key size at v5.2

The host could then check that the key size negotiated by the controller was of a sufficient length to satisfy the requirements of the application or profile using the connection.

There are two issues with this sequence of exchanges.

2.1.4.1 Extra Round Trip

The need to send a R*ead Encryption Key Size* command and receive a *Command Complete* event in response in order that the negotiated key size can be validated by the host, adds a host to controller roundtrip. This will take time and consume energy, albeit small amounts of each.

2.1.4.2 Invalid Key Size

If the key size is invalid for the applications that are to be used between the two devices, the host can determine this and prevent any user data that requires encryption from being transferred. The inability to negotiate a strong key size in the first place leads to a poor user experience.

2.2 About the Host to Controller Encryption Key Control Enhancements

This section explains the main changes relating to host to controller encryption key control in version 5.3 of the Bluetooth core specification.

2.2.1 Capabilities and Benefits

The Bluetooth BR/EDR controller can now inform the host of the negotiated key size at the same time that it informs the host that encryption has been enabled. This eliminates the need for an additional host to controller roundtrip, saving some processing time and associated energy and closes the window during which an unacceptable key size could be in use for a short time.

Establishing an encrypted Bluetooth BR/EDR connection can be a little faster and more efficient as a result.

2.2.2 Technical Highlights

2.2.2.1 Host Controller Interface

Table 3 summarises HCI changes that relate to this aspect of the Bluetooth core specification version 5.3

HCI Command	Description
Encryption Change Event (new version)	V2 of this event now includes a parameter *Encryption_Key_Size* which indicates the size of the key (in octets) which is being used to encrypt the link identified by the *Connection_Handle* response parameter.
Set Min Encryption Key Size command	This is a new HCI command, which the host may use to set the minimum key size which the controller is permitted to accept when negotiating the key size to be used on a new Bluetooth BR/EDR connection. Support for this command is optional.

Table 3 - HCI changes relating to the Encryption Key Control Enhancements

Figure 6 shows the new version of the *Encryption Change Event* in use and the inclusion of the negotiated *encryption_key_size* parameter in its parameter list.

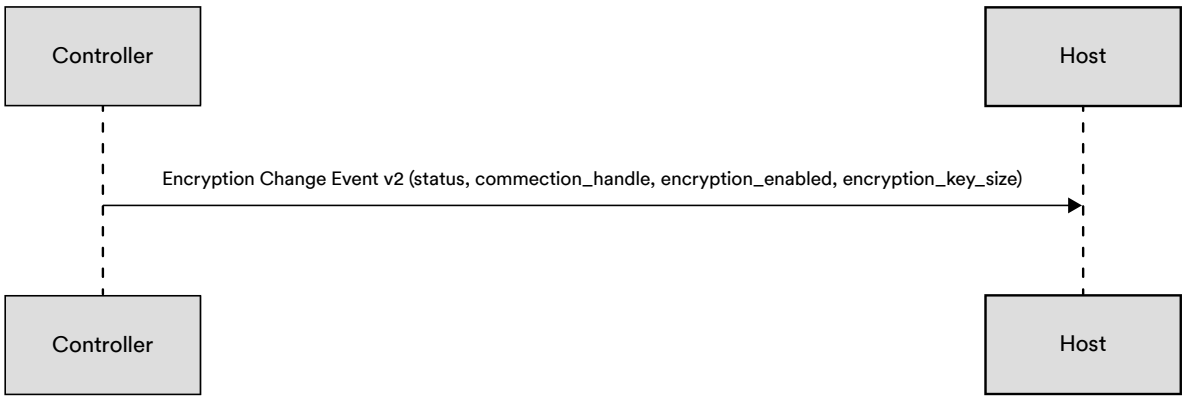

Figure 6 - The enhanced Encryption Change Event

3.0 LE Enhanced Connection Update

3.1 Background

3.1.1 LE-ACL Connections

When two Bluetooth devices are connected, they use either the asynchronous connection-oriented logical transport (LE-ACL or simply ACL) or for applications such as unicast LE Audio, a *Connected Isochronous Stream* (CIS). These and other logical transports are defined in the Bluetooth core specification and constitute part of the Bluetooth data transport architecture.

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*.

When two devices establish an ACL connection, they agree on a number of timing parameters that govern their subsequent communication. Amongst these parameters are *connection interval* and *Peripheral latency*.

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.

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 replies 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. At the start of each connection event, the radio channel to be used is selected.

Figure 7 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.

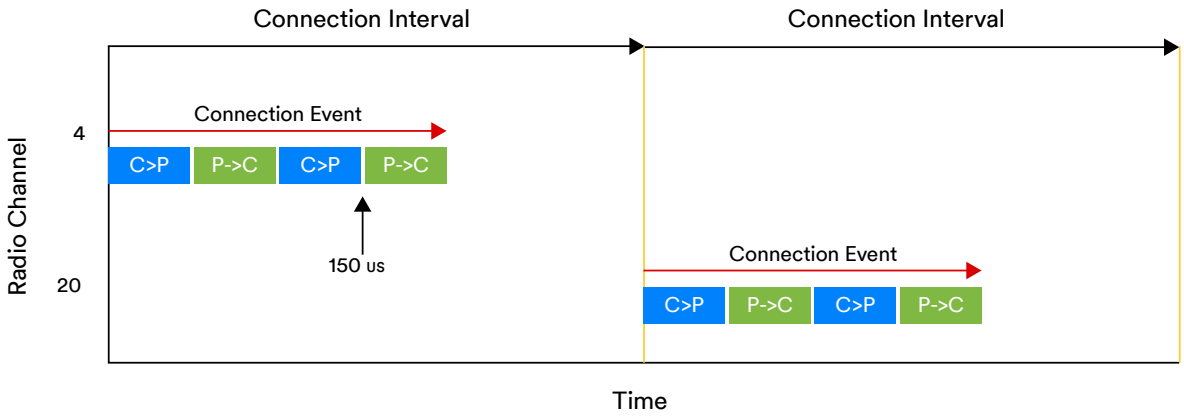

Figure 7 - Basic packet exchange over an LE-ACL connection

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 8 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.

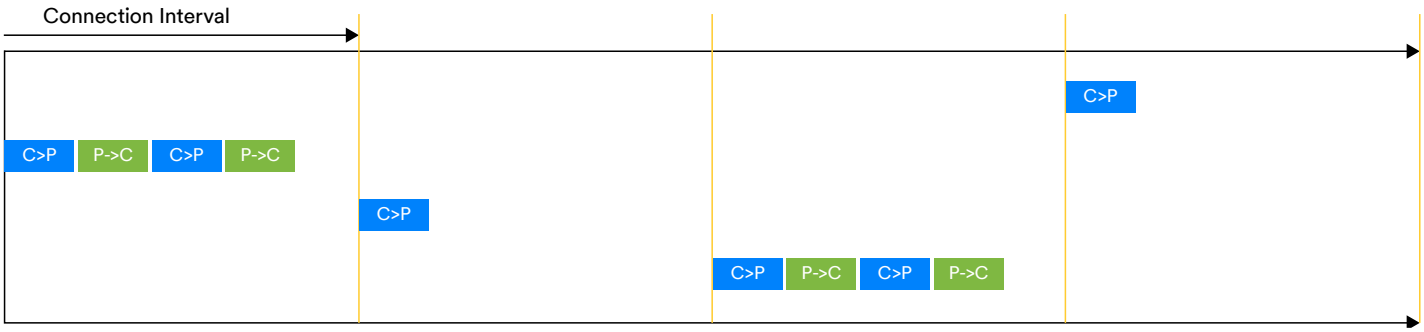

Figure 8 - An ACL connection with Peripheral Latency = 1

3.1.2 Connected Isochronous Streams

A Connected Isochronous Stream (CIS) has a relationship with an ACL connection, which must exist before the CIS can be established. When establishing a CIS, the timing of the first CIS event (known as the *CIS anchor*) is calculated as an offset from the timing of the next ACL event. After this, CIS events occur at intervals known as the ISO *Interval*. Once established, the CIS and associated ACL connection are completely independent of each other and have unique access addresses, identifying them as distinct connections.

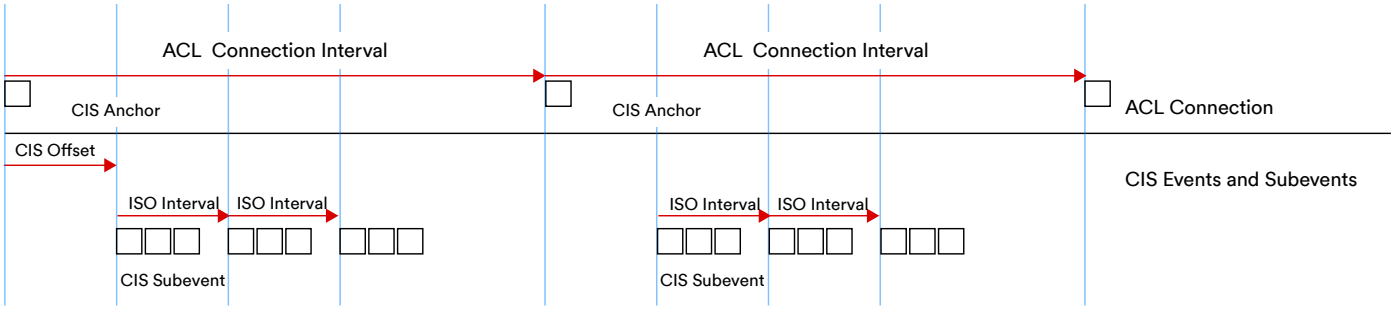

Figure 9 - CIS event timing offset from an ACL event

If the ACL connection associated with a CIS has a long connection interval then it may take significant amount of time to establish the audio stream. This is illustrated in Figure 10.

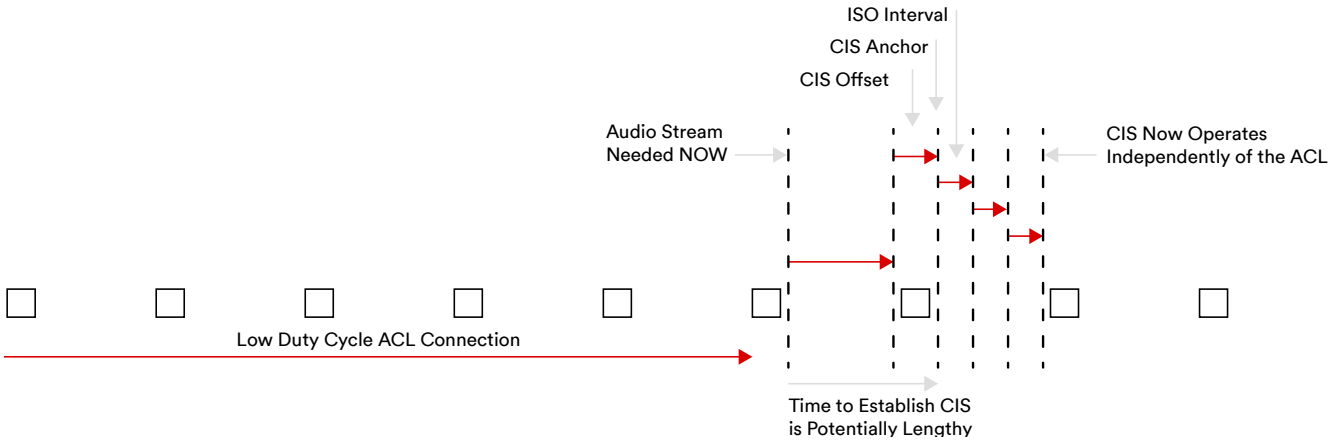

Figure 10 - Time taken to establish a CIS from a low duty cycle ACL connection

3.1.3 Instants

The link layer specification includes a number of control procedures that relate to ACL connections.

Examples include

1. The *Connection Update* procedure which allows connection parameters such as the connection interval to be changed.
2. The Channel Map Update procedure which allows the Central to inform the Peripheral of changes to the channel map which specifies those radio channels that will be available for selection during adaptive frequency hopping and those that will not.
3. The PHY Update procedure which allows either the Central or the Peripheral device to change the transmit or receive PHY or both.

The three selected examples have something in common. All three procedures involve PDUs that include a field called *Instant*.

An instant is a specific connection event, referenced by its connection event counter value and the specification for procedures that use *instants* states that the *Instant field* should be set to the connection event counter value for a connection event that is at least six events (during which the Peripheral will be listening) in the future, relative to the current connection event counter value. Note that the Peripheral's link layer may be subject to a non-zero *Peripheral latency* value and so not necessarily be listening at every connection event.

Link layer procedures that use *instants* generally involve connection property changes which if not executed by both devices successfully, can result in loss of the connection. The concept of an *instant* helps address this risk. Deferring execution of the procedure to a future connection event provides a window during which the Central can retransmit the packet multiple times until an acknowledgement is received from the Peripheral and this significantly improves the probability that the packet will be received. The required procedure is then executed when the connection event indicated in the *Instant* field arises and importantly, it is executed by the initiating device irrespective of whether or not its peer acknowledged the initiating control PDU. This is the reason that such procedures are not executed immediately. They bring about important, potentially connection-breaking changes and the use of instants provides a way for receipt of the control PDU by the Peripheral to be more reliably achieved.

3.1.4 Mixed Duty Cycle Requirements

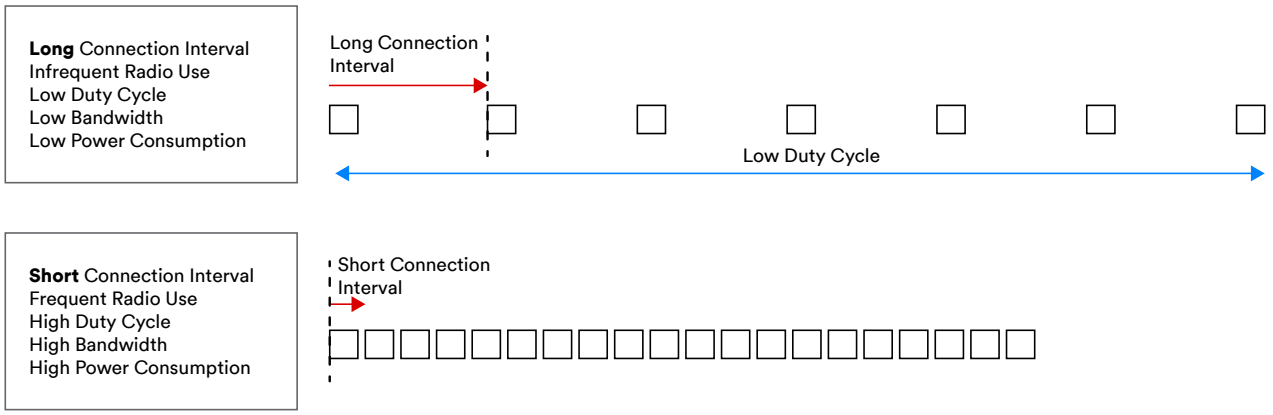

Figure 11 - Duty cycle, bandwidth and power consumption

A low duty cycle connection uses little power and is achieved through the use of a long *connection interval* and perhaps also, a non-zero *Peripheral latency* value.

A high duty cycle connection will use more power but provide higher bandwidth and is achieved through the use of a short connection interval and a *Peripheral latency* value of zero.

It can sometimes be necessary for a device to use a low duty cycle connection for some use cases but be able to switch up to a high duty cycle when required to support other use cases.

As an example, consider a hearing aid which is paired with the user's smartphone. Most of the time the hearing aid does not need to communicate with the smartphone and simply amplifies sounds from the immediate environment. But if the user decides to play some music or receives a call on their smartphone, audio must be delivered over a Bluetooth connection to the hearing aid.

The hearing aid is battery powered and the battery must last as long as possible. When it becomes necessary to handle audio from the smartphone, there must be no noticeable delay in preparing the Bluetooth connection for the change in use case.

One strategy that can be used to reduce the time it takes to prepare a Bluetooth connection and audio stream between the hearing aid and smartphone is to maintain a persistent connection all of the time so that it is already available when required. Most of the time the connection will not be actively used and so to save battery power the connection will initially be established with a very low duty cycle. When audio must be delivered over the connection, the connection's parameters will be changed so that a high duty cycle is established, providing the higher bandwidth needed for the new use case.

Figure 12 illustrates a connection update procedure changing the connection interval from a relatively long value to a short value and as such, from a low duty cycle connection to a high duty cycle. Note the inclusion of an *instant* which specifies that connection event #10 is the event at which the connection update must be applied by both Central and Peripheral.

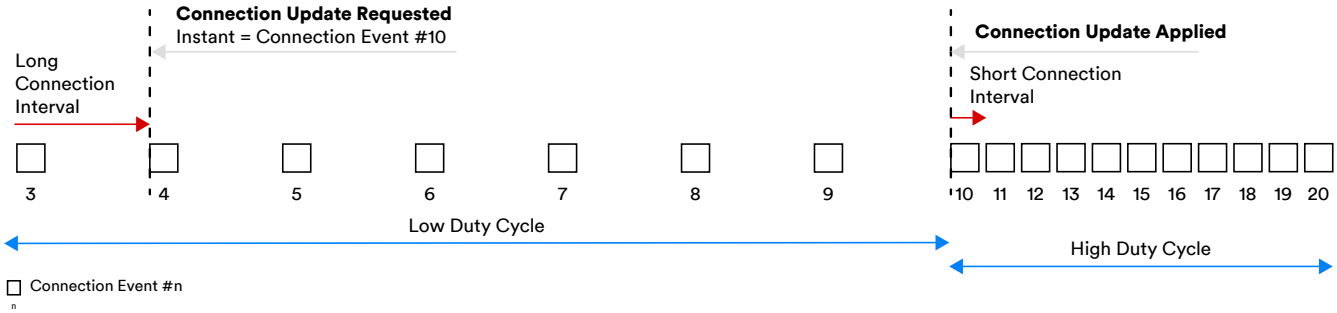

Figure 12 - A connection update procedure changing the connection from a low to a high duty cycle

But changing the connection interval and Peripheral latency parameters of an existing connection can take a significant amount of time.

To start the *connection update* procedure, the Host must send the controller a *LE Connection Update* command. The controller must respond by initiating the link layer Connection Update procedure, sending a LL_CONNECTION_UPDATE_IND PDU to the Peripheral with the *Instant* field specifying a connection event which is *at least six events in the future* at which point the connection parameter changes must be applied by both the Peripheral and Central.

Having received a connection update request, the Peripheral will take a significant elapsed time to apply it for the following reasons:

1. The minimum number of connection events which need to elapse, as specified using the *Instant* field should be six (where the Peripheral is listening) according to the Bluetooth core specification.
2. Connection events in this low duty cycle connection occur relatively infrequently due to the large *connection interval* value that is in use. A *connection interval* of one second, with an *Instant* specified six events in the future will take six seconds to occur, for example.
3. The number of connection events which must elapse may be increased by a multiple of (*Peripheral latency* + 1)

Consequently, the total time which may need to elapse before the connection is updated can be quite substantial and at odds with the need to provide a suitable user experience.

It is this issue which gave rise to the decision to provide an enhanced connection update capability in the Bluetooth core specification version 5.3.

3.2 About the LE Enhanced Connection Update Change

This section explains the LE Enhanced Connection Update change made in version 5.3 of the Bluetooth core specification.

3.2.1 Capabilities and Benefits

The LE Enhanced Connection Update change introduces the concept of *subrated connections*. A *subrated connection* achieves a power saving low duty cycle in a different way to non-subrated connections and is able to switch up to a high duty cycle more quickly. Subrated connections are also able to handle variable packet rates or *bursty* traffic in an efficient way.

The ability to switch quickly from a low duty cycle connection to a high duty cycle connection when required and in a short time, will provide an improved user experience for some important product types and is the main benefit of the new *subrated connections* feature.

3.2.2 Use Cases

Use cases which are expected to benefit from subrated connections include Bluetooth LE Audio products such as hearing aids and monitoring systems which use sensors.

An audio device like a hearing aid can maintain a subrated connection to a smartphone, making the connection available for immediate use when it becomes necessary to handle audio communication. Subrating saves power but also allows the connected isochronous stream to be established very quickly.

Using subrated connections, a sensor can efficiently maintain a persistent, low duty cycle subrated connection to another device which collects sensor data. When

required it can then switch the connection up to a high duty cycle quickly so that accumulated sensor data can be uploaded over a higher bandwidth connection.

3.2.3 Technical Highlights

3.2.3.1 Connection Subrating

i) Basic Principles

Subrated connections are 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*. ACL connection properties which also have a significant effect on the behaviour of a subrated connection are the *connection interval* and the *Peripheral latency* value.

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. Connection events are actively used when the peripheral listens to the event or sends data in the event. A subrated connection can therefore have a short ACL connection interval but still exhibit a low duty cycle.

Figure 13 illustrates the basic concepts relating to subrated connections.

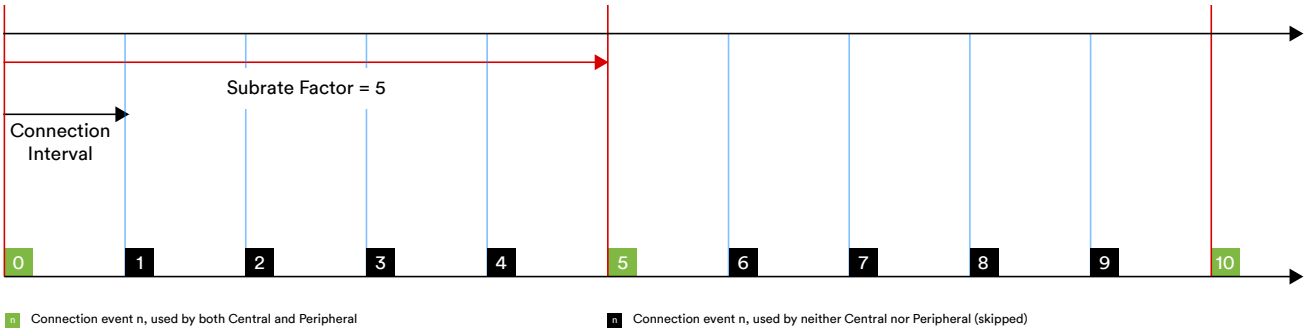

Figure 13 - A basic subrated connection with subrate factor=5

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.

ii) Subrate Base Events

A subrated connection defines a number of consecutive ACL connection events of which one will be actively used (the subrated connection event). But the subrated connection event does not have to be the first ACL connection event in the sequence of events. The choice of connection event to use from within the sequence of subrate factor events is controlled by the subrate base event value which is used to determine the phase of the subrated events. Figure 14 shows a subrated connection where the phase of the subrated events has been shifted by two connection events by specifying a subrate base event value of 2.

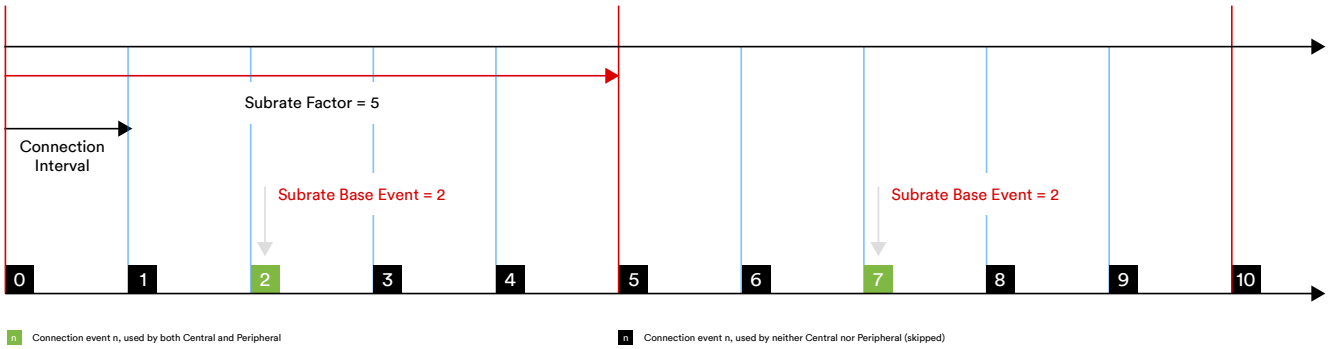

*Figure 14 - A subrated connection with subrate factor=5, **subrate base event=2** and Peripheral latency=0*

The subrate base event property provides packet schedulers with control over connection events and radio use so that a workable schedule can be established across one or more concurrent connections.

iii) Continuation Events

The rate of packets exchanged over a connection and the way in which the rate varies is called the *traffic profile*. In some cases, the traffic profile will be flat with a steady packet rate. Perhaps 20 packets are exchanged at every connection event under all circumstances for example. But in other cases, the nature of the application might mean that the packet rate is variable, with a low packet rate some of the time and then short bursts of larger numbers of packets transmitted in the same connection event at others.

Connection subrating uses the concept of *continuation events* so that when a subrated connection is actively in use, evidenced by having handled a link layer packet which has a non-zero length during the current subrated connection event, a specified number of the following ACL connection events will be selected for use rather than those events being skipped as would otherwise have been the case. This ensures that bursts of activity from the application are not slowed down by subrating. The subrated connection adjusts its effective connection rate according to its properties and the level of activity over the connection.

After an actively used subrated connection event, a number of the connection events that follow on immediately from the subrated connection event will also be selected for use. These events are referred to as *continuation events*. How many continuation events are created is determined by the value of the *continuation number* property of the

connection. For example, if *continuation number* is 2, then after an actively used subrated connection event, the next two ACL connection events will not be skipped.

Furthermore, if there is activity in a *continuation event* this causes the next *continuation number* connection events to be actively used. In this way, a sliding window of usable connection events is created. The event continuation process ends at the next subrated connection event, where the process then restarts.

Figure 15 and Figure 16 illustrate two scenarios involving continuation events. In Figure 15, *continuation number* has a value of 1 whereas in Figure 16 *continuation number* is set to 2.

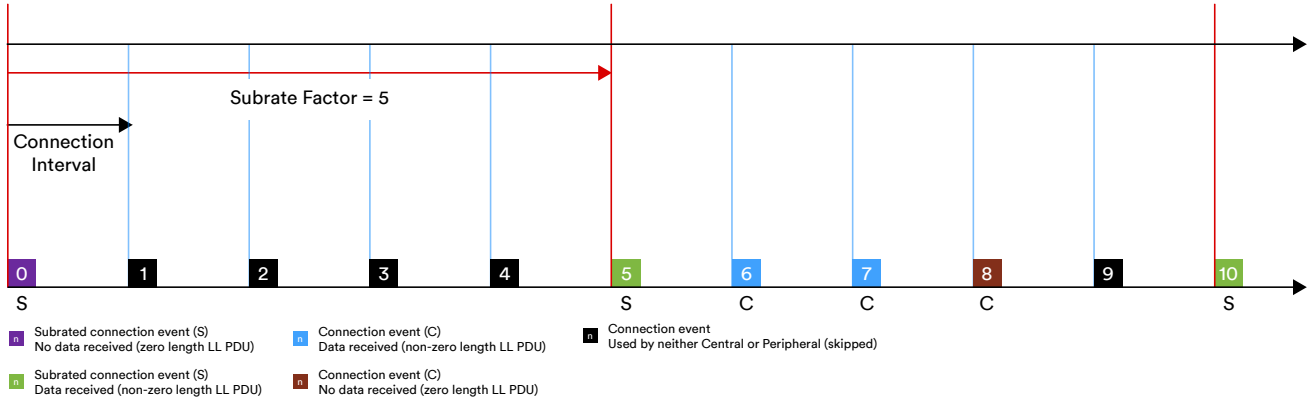

Figure 15 - Continuation events in a subrated connection. ***Continuation Number=1***

In Figure 15 the subrated connection running over the ACL connection has a *continuation number* property with a value of 1 and a number of important points are illustrated.

- At connection event #0 (the subrated connection base event), the link layer packet received had no payload and so continuation was not warranted. All other connection events until the next subrated connection event are therefore skipped.
- At the subrated event which occurs at ACL connection event #5, at least one non-zero length link layer packet was transmitted and so the next connection event (event #6) is designated a *continuation event* and is actively used.
- At least one non-empty link layer packet was handled during this continuation event, connection event #6 and so the next connection event, event #7 also becomes a continuation event.
- Data is handled in event #7 and so event #8 becomes a continuation event too.
- During event #8, there are no non-empty packets and so the sequence of continuation ends here.

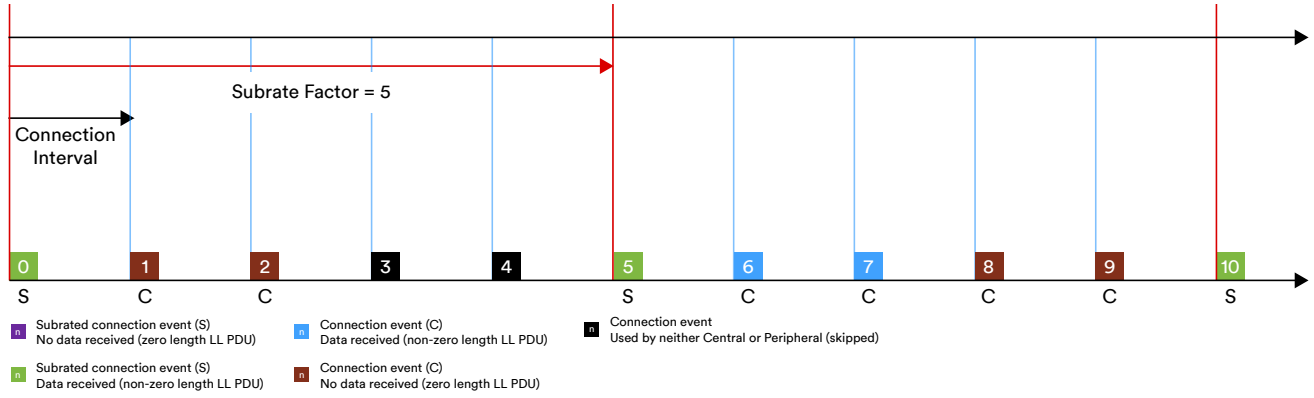

Figure 16 - Continuation events in a subrated connection. ***Continuation Number=2***

In Figure 16 continuation number has a value of 2 and the following points are illustrated.

- At connection event #0, one or more non-zero length packets are handled. This causes the next two connection events to be designated continuation *events* since *continuation number* is 2.
- Neither of the *continuation* events #1 and #2 involve any user data and so the continuation window ends at connection event #2
- At the next subrated connection event (connection event #5) user data is handled. This causes events #6 and #7 to be continuation events.
- Data is handled in continuation event #6 so the continuation window is extended, with connection event #8 becoming designated a continuation event.
- Data is also handled in continuation event #7, so connection event #9 is also a continuation event.
- No user data is handled in either continuation event #8 or #9. A new subrated connection event starts at event #10 so for both of these reasons, the continuation window rooted in subrated connection event #5 ends. The subrated connection event #10 involves user data and so events #11 and #12 become continuation events (not shown).

iv) Peripheral Latency and Subrated Connections

The effect of the *Peripheral latency* connection property on non-subrated ACL connections was explained in section 3.1.1 LE-ACL Connections. When used with a subrated connection however, the effect can sometimes differ in a number of ways, depending on whether or not continuation events are in use and the activity or absence of activity over the connection.

When Peripheral latency is greater than zero in a subrated connection, the Peripheral latency concept applies to subrated connection events only and these occur at the intervals determined by the subrate factor value. Figure 17 shows the case of a subrated connection which has a subrate factor of 3, a continuation number set to 0 and a Peripheral latency set to 1. Every other subrated connection event is skipped due to the Peripheral latency value of 1 in conjunction with the subrate connection parameters.

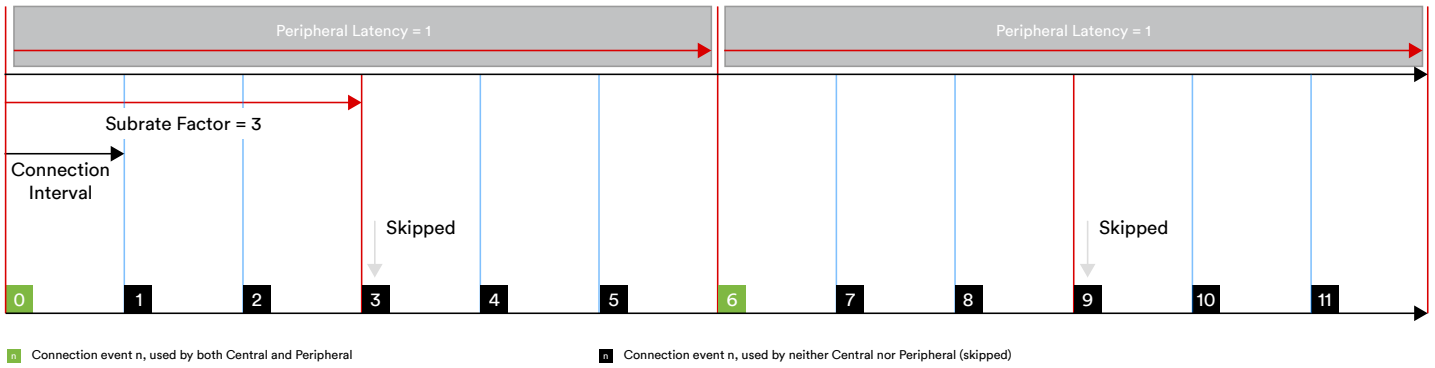

*Figure 17 - A subrated connection with **Peripheral Latency=1**, subrate factor=3 and continuation number=0*

Figure 18 shows the effect of a *non-zero continuation number* when used with a non-zero *Peripheral latency* value. The Peripheral listens at every other subrated connection event as you would expect but the *Peripheral latency* property value has no effect on the behaviour during continuation events. After receiving data at a subrated connection event, the Peripheral listens at the next continuation event exactly as would have been the case with a *Peripheral latency* value of zero.

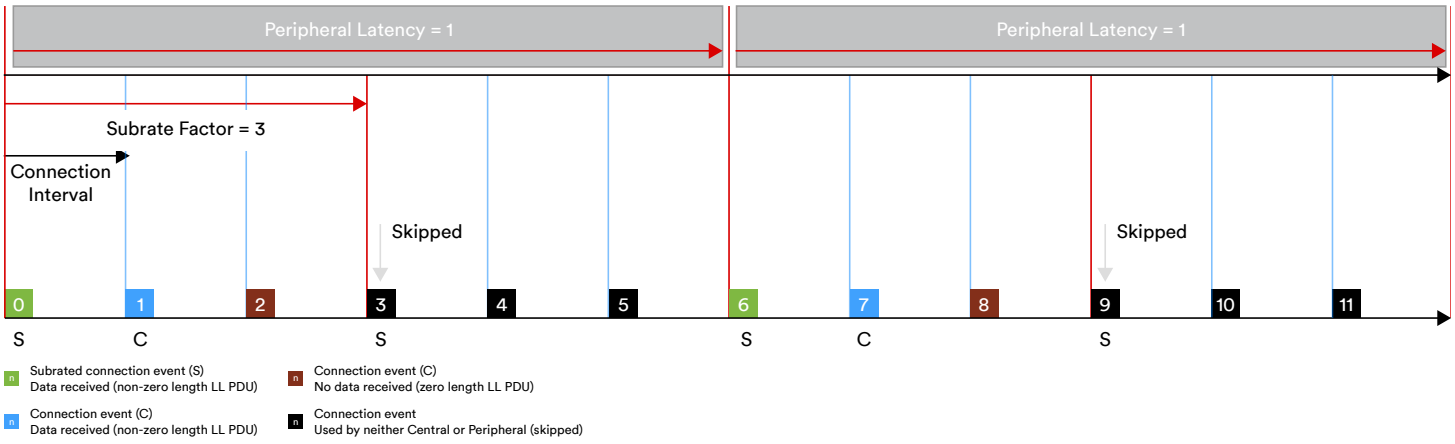

*Figure 18 - A subrated connection with **Peripheral Latency=1**, subrate factor=3 and **continuation number=1***

If no valid packets are received when a Peripheral using a non-zero *Peripheral latency* value listens during a subrated connection event, rather than skip the next (*Peripheral latency* - 1) subrated events, the Peripheral should listen at every subsequent subrated connection event until at least one valid packet is received (note though that the choice permitted here is an implementation decision). When this occurs, the Peripheral is not required to listen at the following subrated connection events, per its *Peripheral latency* value.

Figure 19 and the following section show an example of a Peripheral that applies Peripheral latency, when allowed, with a series of 23 connection events over a subrated connection. The *subrate factor* value is 3, Continuation Number is 0, and it has a *Peripheral latency* value of 1. The following points are depicted in this illustration:

- At event #0 (a subrated connection event), the Peripheral listens but did not receive a valid packet.
- Consequently, instead of skipping the second subrated connection event (event #3), the Peripheral listens. Again, no valid packet is received.
- The Peripheral listens again at event #6 and this time receives one or more valid packets. There is activity across this connection now so the *Peripheral latency* value of 1 is applied by the Peripheral and it skips the next subrated connection event, event #9.
- At event #12, the Peripheral listens again but receives no valid packets. Consequently, the effect of *Peripheral latency* is suspended and the Peripheral listens at the next subrated connection event.
- Listening at event #15, the Peripheral receives a valid packet. *The Peripheral latency* effect is reinstated because of this activity and the next subrated connection event, event #18 is skipped.
- The Peripheral listens at event #21 as expected and receives a valid packet. Given this activity, the *Peripheral latency* will be applied and the next subrated connection event (not shown) will be skipped.

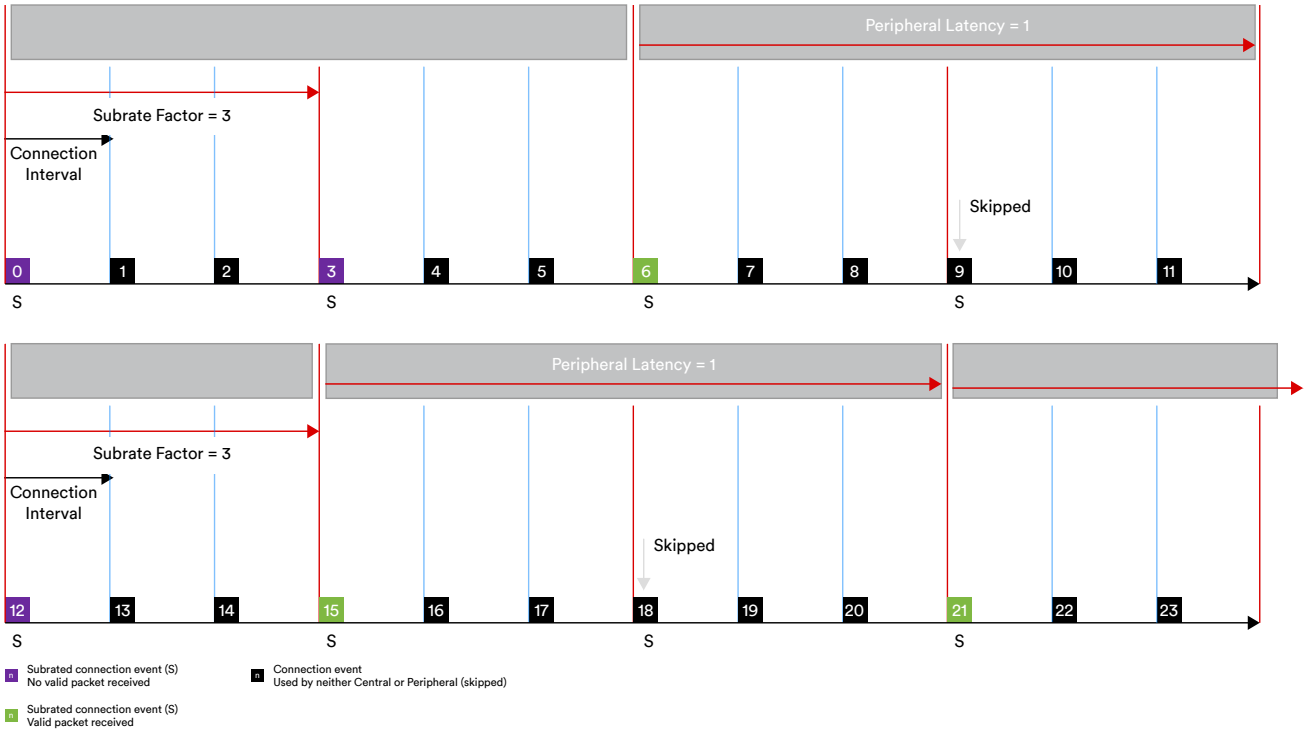

Figure 19 - Peripheral latency ignored during periods of inactivity

v) Connected Isochronous Streams and the Effect of Subrating

Subrating does not affect the way in which connected isochronous streams work. When establishing a CIS however, this is always done with reference to an existing ACL connection. If that connection is a subrated connection, it can have a power saving low effective duty cycle and a short connection interval. This means that the time to the next ACL event from which the CIS events are ultimately scheduled is also short and this in turn means that the audio stream can be established very quickly, delivering an improved user experience but without the energy cost that a non-subrated, high duty cycle connection would otherwise incur.

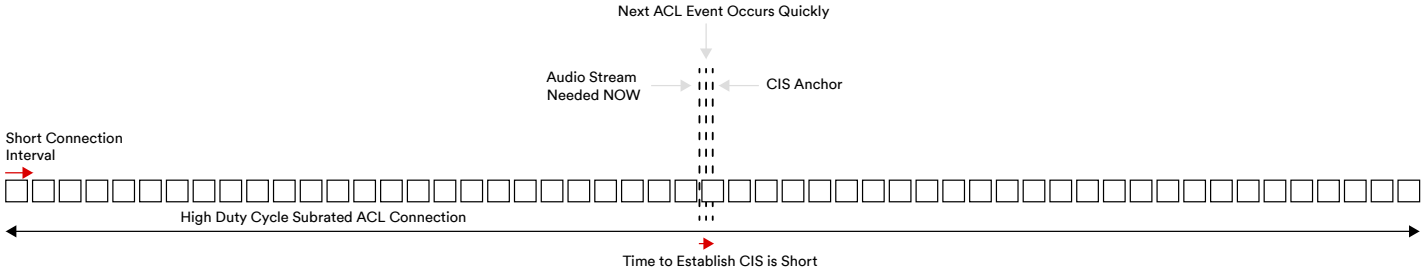

Figure 20 - A subrated connection allows rapid CIS establishment and low power consumption to coexist

3.2.3.2 Subrate Connection Updates

3.2.3.2.1 The Link Layer Subrate Update Procedure

The Link Layer Connection Subrate Update procedure allows the Central device to inform the Peripheral that the subrate connection properties must change. The Central will initiate this procedure in response to a request from its own Host over HCI or from the remote Peripheral in the form of a LL_SUBRATE_REQ PDU (see 3.2.3.2.2 The Link Layer Subrate Request Procedure).

The procedure starts with the Central sending a LL_SUBRATE_IND PDU to the Peripheral containing values for the following connection parameters:

- Subrate Factor
- Subrate Base Event
- Peripheral Latency
- Continuation Number
- Supervision Timeout

On sending the LL_SUBRATE_IND PDU, the Central device is said to have entered *subrate transition mode*. On receiving an acknowledgment to the LL_SUBRATE_IND PDU, the Central exits subrate transition mode and switches to using the new subrate connection parameters.

On receiving a LL_SUBRATE_IND PDU, the Peripheral will switch to using the new subrate connection parameters immediately.

1. Those events defined by the current subrate factor and subrate base event values and
2. Those events defined by the new subrate factor and subrate base event values that have been requested.

Retransmissions are performed to maximise the probability that the PDU will be received by the Peripheral during the supervision timeout period. They are sent during both the old and new sets of subrate events so that if the Peripheral receives the PDU and adopts the new parameters but its acknowledgement is not immediately received by the Central, retransmissions from the Central should be received by the Peripheral during the new subrate connection events and acknowledged again.

3.2.3.2.2 The Link Layer Subrate Request Procedure

The Link Layer Subrate Request Procedure allows the Peripheral to ask the Central to change some of the subrate connection parameters. When requested to do so by its Host, the Peripheral sends a LL_SUBRATE_REQ to the Central, indicating minimum and maximum requested subrate factor values, a maximum requested Peripheral latency value, a continuation number, and a new supervision timeout value. The Central evaluates the request, taking into consideration packet scheduling issues and the range of acceptable subrate parameters that may have been specified by its Host and if the request can be accommodated, initiates the Subrate Update Procedure.

3.2.3.3 Host Controller Interface Changes

Two new HCI commands and one new event have been added to the Bluetooth core specification version 5.3 to support the LE Enhanced Connection update change and these are summarised in Table 4.

3.2.3.2.2 The Link Layer Subrate Request Procedure

The Link Layer Subrate Request Procedure allows the Peripheral to ask the Central to change some of the subrate connection parameters. When requested to do so by its Host, the Peripheral sends a LL_SUBRATE_REQ to the Central, indicating minimum and maximum requested subrate factor values, a maximum requested *Peripheral latency* value, a continuation number, and a new supervision timeout value. The Central evaluates the request, taking into consideration packet scheduling issues and the range of acceptable subrate parameters that may have been specified by its Host and if the request can be accommodated, initiates the Subrate Update Procedure.

3.2.3.3 Host Controller Interface Changes

Two new HCI commands and one new event have been added to the Bluetooth Core Specification version 5.3 to support Connection Subrating and these are summarised in Table 4.

HCI Command	Description
LE Set Default Subrate command	Allows a Central's Host to specify to its Controller, the subrate connection parameter values and value ranges that a connected Peripheral may request.
LE Subrate Request command	Allows the Host of either the Central or Peripheral device to request changes to the subrate parameters that apply to a current connection.
E Subrate Change Event t	Used by a controller to inform its host of changes to subrate connection parameters that have been made.

Table 4 - HCI changes relating to LE Enhanced Connection Update

3.2.3.4 Subrate Updates in Action

Figure 21 shows an example message sequence which starts with a subrate request having been issued by the Central device's Host, progresses through the Subrate Update procedure, with the Central in subrate transition mode and ends with both Central and Peripheral Hosts being notified with an HCI_LE_Subrate_Change event that the change has been made. At several points, communication problems are causing CRC failures on packets received and packet retransmissions are taking place as a result.

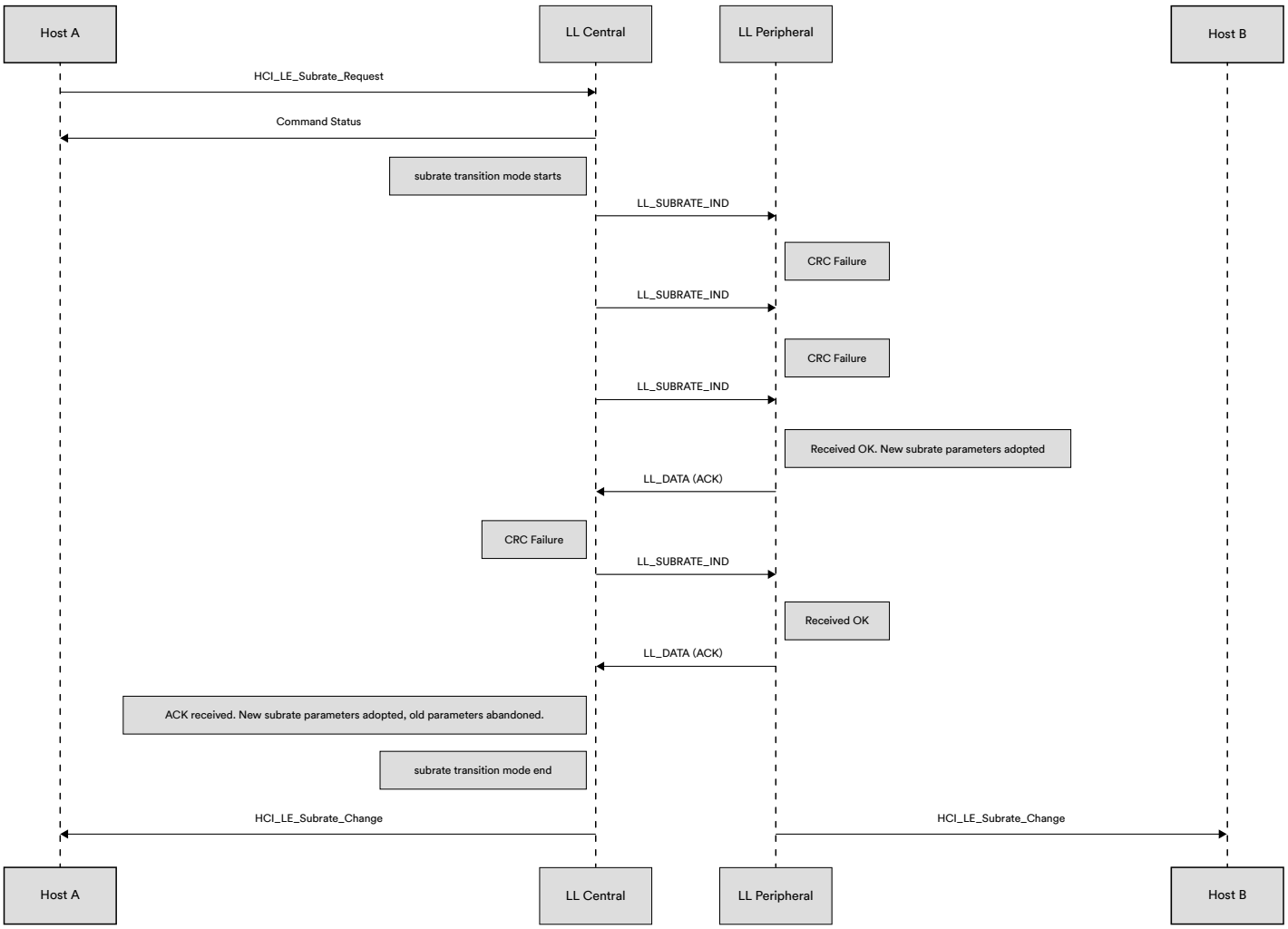

Figure 21 - Example subrate update procedure message sequence

Figure 22 shows how connection events are used during and after subrate transition mode. The connection has a subrate factor value of 5 at the start and the subrate update procedure has been initiated with a new subrate factor of 2 having been specified. The subrate base event is not changed.

From event #0, up to and including to event #14, the Central device is in subrate transition mode and makes use of both those subrated connection events that correspond to the old (current) subrate factor value of 5 and the new subrate factor, which is requested, with a value of 2. At event #15, the Central has received an acknowledgment to its LL_SUBRATE_IND PDU and so subrate transition mode ends, with the Central adopting the new subrate connection parameters. From that point on, the subrate factor value of 2 applies only.

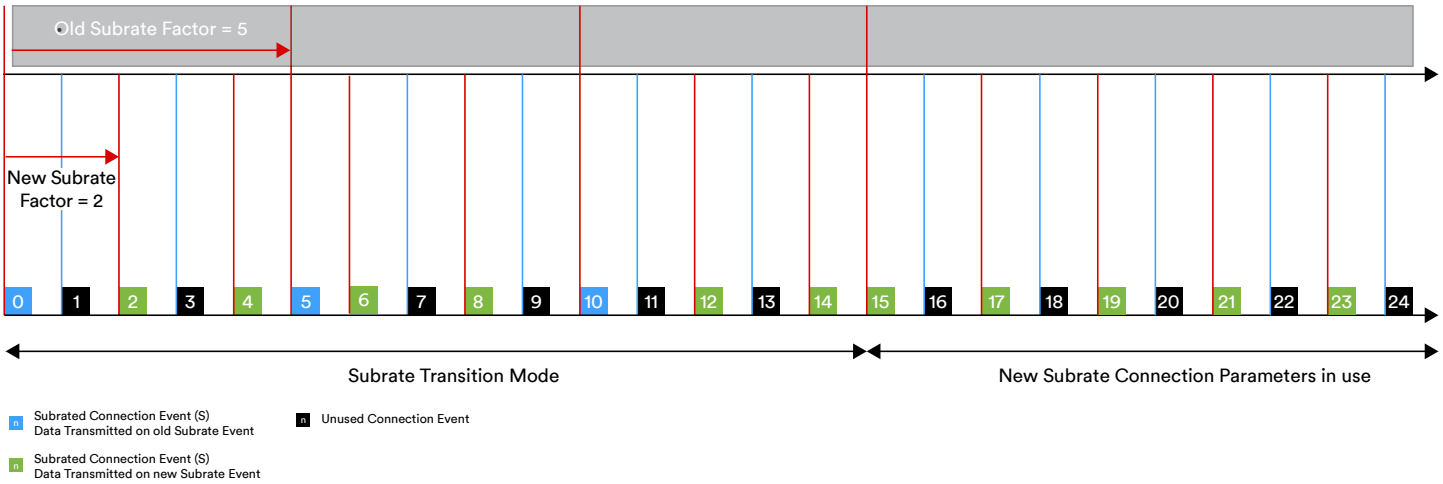

Figure 22 -Connection event use and subrate transition mode

3.2.3.5 Subrate Updates vs Connection Updates

This section provides a recap of the benefits of subrate connection updates, how they differ from standard ACL connection updates and how they deliver their primary benefits.

Subrated connections allow changes from a low duty cycle connection to a high duty cycle connection to be made quickly. ACL connection updates are slow to complete in comparison to subrate connection updates because of their use of Instants. Instants allow the Central to give the Peripheral advanced notice of the change to connection parameters, which will be made by the Central irrespective of whether or not an acknowledgement is received from the Peripheral. Scheduling the connection update to take place at a particular connection event in the future gives the Central a window of time during which to retransmit the connection update request PDU if necessary. Instants are used because of the potential for a connection update to break the connection. This was explained in section 3.1.3 Instants.

Subrate connection changes do not risk breaking the connection and so the link layer subrate update procedure does not make use of Instants. Changes to subrate parameters are made immediately by the Peripheral when received and immediately by the Central on receiving an acknowledgement from the Peripheral. The delay caused by the window associated with Instants is not a factor in the time it takes to make subrate connection updates and hence these types of change can be made quickly.

Subrated connections have both an ACL connection interval and an effective connection interval and this contributes to the shorter time it takes to apply a change to subrated connection parameters whilst allowing a low duty cycle to be retained when necessary.

4.0 LE Channel Classification

4.1 Background

In this section, aspects of Bluetooth as defined in version 5.2 of the core specification that have relevance to the topic of channel classification are presented.

4.1.1 Radio Channels

Bluetooth uses the 2.4GHz (ISM) radio band. Bluetooth BR/EDR divides the ISM band into 80 channels, each 1 MHz wide and Bluetooth LE divides the band into 40 channels of 2 MHz width.

Each radio channel is defined by an upper and lower frequency. Figure 23 shows the radio channels used by Bluetooth LE

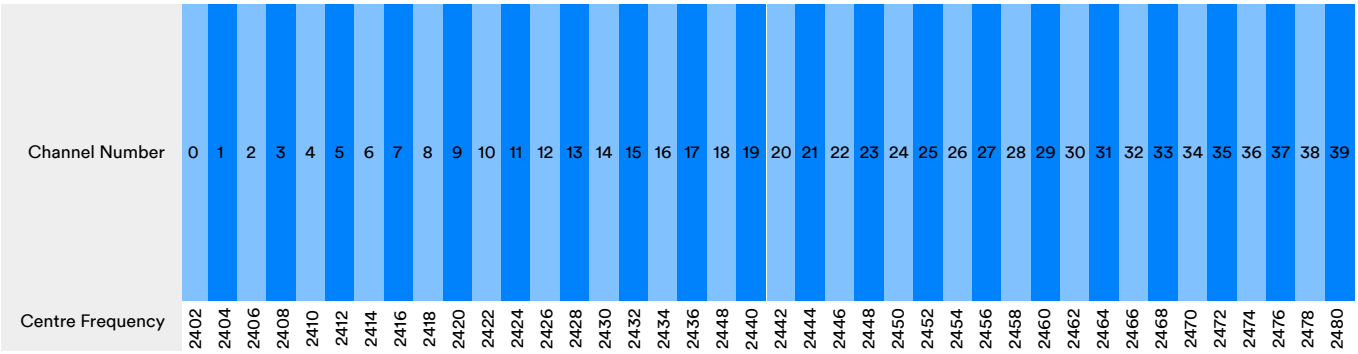

Figure 23 - Bluetooth LE and radio channels in the ISM band

4.1.2 Adaptive Frequency Hopping

When two devices are connected, a spread spectrum technique called adaptive frequency hopping (AFH) is used and the radio channel used for packet communication changes at 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*. Table 5 explains each classification.

Channel Classification	Meaning
Used	The channel is suitable for use.
Unused	The channel should not be used.

Table 5 - Channel Map Classifications

Figure 24 provides an example of AFH in action and was captured using a protocol analyser during testing. The histogram shows the number of packets sent on each of the used channels when performing AFH.

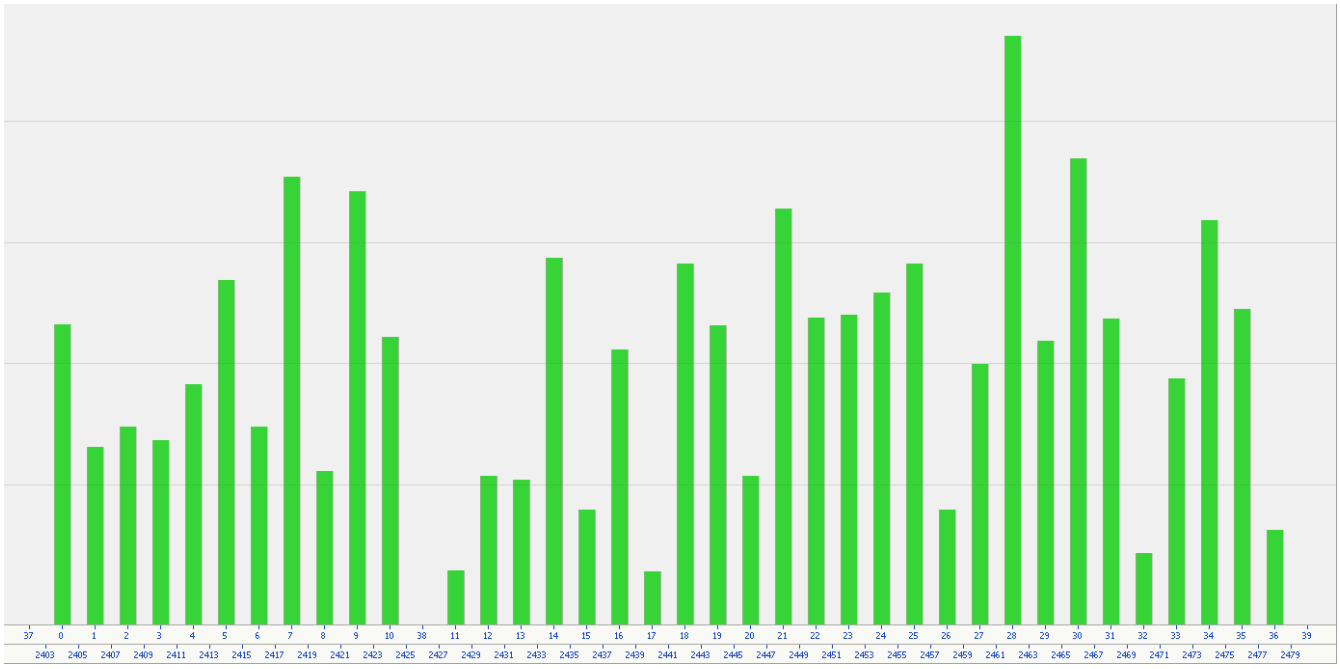

Figure 24 - A histogram of channel use over time

Up to version 5.2 of the Bluetooth core specification, channel classification in Bluetooth LE is performed by the Central device only and the Central's controller can use locally acquired measurements of channel performance or information provided by the host in determining the channel classifications.

4.1.3 Channel Map Updates

The performance of radio channels can change, perhaps as other devices in the environment come and go. Consequently, channel classification may be updated by the Central's controller when necessary.

The Peripheral needs to work from the same channel map as the Central device and so if the Central's controller updates the channel map, it executes the *channel map update procedure* which involves sending the new channel map to the Peripheral device in an LL_CHANNEL_MAP_IND PDU.

4.2 About the LE Channel Classification Change

This section explains the changes that have been made to the way in which channel classification is performed by Bluetooth LE in version 5.3 of the core specification.

4.2.1 Capabilities and Benefits

Prior to version 5.3 of the core specification, Bluetooth LE channel classification was performed only by the Central device. However, if two connected devices are not in close proximity with each other, the radio conditions of one may differ from those experienced by the other. Consequently, when channel classification is performed only by the Central device, it is possible that the channel map can contain channels classified as *used* which are not a good choice for the remote Peripheral. This can result in packet collisions and generally degraded throughput and reliability.

> *Note: It has always been possible for **Bluetooth BR/EDR** channel classification to be performed by both peer devices in a connection*

Channel classification in Bluetooth LE may now be performed by the Peripheral device as well as the Central. The Peripheral can report its channel classifications to the connected Central device so that this data may also be considered when updating the Central's channel map. This can result in only channels that are performing well for both devices being used in AFH and consequently improve throughput and reliability through reducing the number of collisions that might otherwise have occurred.

4.2.2 Use Case

Two friends, Alice and Bob are using their Bluetooth headphones to simultaneously listen to music playing on Alice's smartphone. Alice's smartphone is very new and supports the latest version of Bluetooth, version 5.3. Both friends have the same make and model of Bluetooth headphones but Bob's headphones are a year old and Alice's are brand new. As far as the two friends are concerned their headphones are identical.

The headphones are connected to the smartphone with each pair acting as a Peripheral and the smartphone acting as the Central device.

The headphones and smartphone are using LE Audio technology and so Bluetooth LE is transporting encoded audio data between the devices. Thanks to the LC3 codec, the audio quality is excellent. However, with the smartphone left sitting on a table, Bob notices that when he walks some distance away from it towards the kitchen the quality of the audio that he hears often degrades at some point and he is puzzled because his headphones and smartphone are comfortably in range of each other.

Alice and Bob are equally curious about this unexpected phenomena and so Alice conducts the same experiment, walking away from the smartphone towards the table. The audio quality does not degrade for Alice and she can walk all the way into the kitchen, still enjoying high quality audio with no evident issues.

Bob and Alice each consult the documentation that came with their headphones, scouring the small print carefully and Alice spots some important details. It turns out that her headphones are a second edition of this make and model but Bob's are from the

first edition. And more to the point, Bob's headphones support Bluetooth 5.2 whilst Alice's support Bluetooth 5.3. This must be something to do with the difference in behaviour that they each experience, they correctly reason.

What neither of them know is that by walking away from the table towards the kitchen, they move into range of a neighbour's Wi-Fi. Alice's Bluetooth 5.3 headphones report their local classification of Bluetooth channels back to the smartphone and so those Bluetooth channels that are impacted by the neighbour's Wi-Fi are marked as unused for this connection and are therefore avoided. But Bob's older Bluetooth 5.2 headphones cannot do this and channel classification is performed by the smartphone only for his connection. Since the smartphone is not in range of the neighbouring Wi-Fi network, those Bluetooth channels which are being interfered with near to the kitchen are classified as *used* for the connection to Bob's headphones and consequently he experiences interference whilst Alice does not.

Bob makes a mental note to treat himself to new headphones with Bluetooth 5.3 for his next birthday!

4.2.3 Technical Highlights

4.2.2.1 Link Layer Changes

A bit from the link layer FeatureSet bit mask has been allocated for indicating whether or not LE channel classification is supported.

The Central device's link layer may enable LE channel classification in the remote Peripheral by performing the new Channel Classification Enable procedure. This involves the Central sending a LL_CHANNEL_REPORTING_IND PDU to the Peripheral. This PDU contains a flag which indicates whether channel reporting should be enabled or disabled and some timing parameters which the Peripheral will use to set the reporting schedule.

When enabled by the Central device, the Peripheral performs the link layer channel classification reporting procedure. When there has been a change in the Peripheral's channel classification then subject to the timing parameters specified by the Central device, the Peripheral sends a LL_CHANNEL_STATUS_IND PDU to the Central.

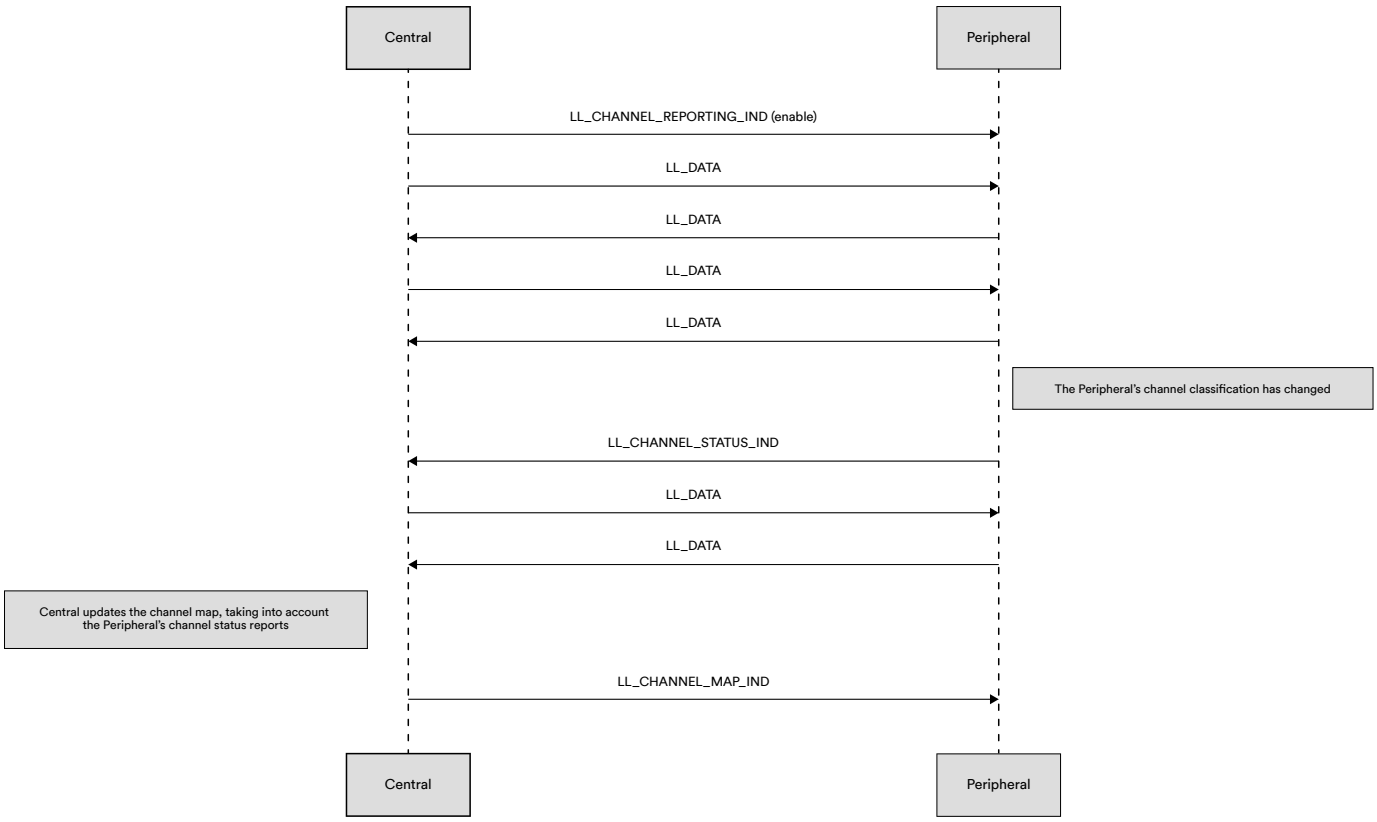

Figure 25 - Peripheral reporting channel classification changes to the Central

The LL_CHANNEL_STATUS_IND PDU contains an array of thirty-seven 2-bit channel classification values, one for each of the general data channels. Defined classification values and their meaning are shown in Table 6.

Channel Classification Value	Meaning
0b00	unknown
0b01	good
0b10	reserved for future use
0b11	bad

Table 6 - Channel Classification Values

5. Conclusion

Bluetooth core specification version 5.3 adds a number of new features which collectively will improve reliability, energy efficiency and user experience in many types of product. In addition, the Alternate MAC and PHY (AMP) feature has been removed.

References

Item	Location
Bluetooth Core Specification v5.3	https://www.bluetooth.com/specifications/specs/
The Bluetooth LE Security Study Guide	https://www.bluetooth.com/bluetooth-resources/le-security-study-guide/
The Bluetooth Security and Privacy Best Practices Guide (SIG members only)	https://www.bluetooth.com/bluetooth-resources/bluetooth-security-and-privacy-best-practices-guide/
Bluetooth Core Specification Version 5.0 Feature Overview	https://www.bluetooth.com/bluetooth-resources/bluetooth-5-go-faster-go-further/