Bluetooth Core Specification v5.1

Feature Overview

Bluetooth Core Specification v5.1 contains a series of updates to the *Bluetooth*® core specification. This document summarizes and explains each change.

Bluetooth Core Specification v5.1 should be consulted for full details.

Author: Martin Woolley

Version: 1.0.1

Revision Date: 9 December 2020

Revision History

Version	Date	Author	Changes
1.0.0	28 January 2019	Martin Woolley	Initial Version
1.0.1	9 December 2020	Martin Woolley	Language Changes

Table of Contents

1.0 Direction Finding

Overview

Bluetooth proximity solutions and positioning systems currently use signal strength to estimate distance. A new direction finding feature in Bluetooth Core Specification v5.1 makes it possible for Bluetooth devices to determine the direction of a Bluetooth signal transmission.

This new feature offers two different methods for determining the angle that a Bluetooth signal is being transmitted from with a high degree of accuracy. The two methods are called Angle of Arrival (AoA) and Angle of Departure (AoD).

Each of the techniques requires one of the two communicating devices to have an array of multiple antennae, with the antenna array included in the receiving device when the AoA method is used and in the transmitting device when using AoD.

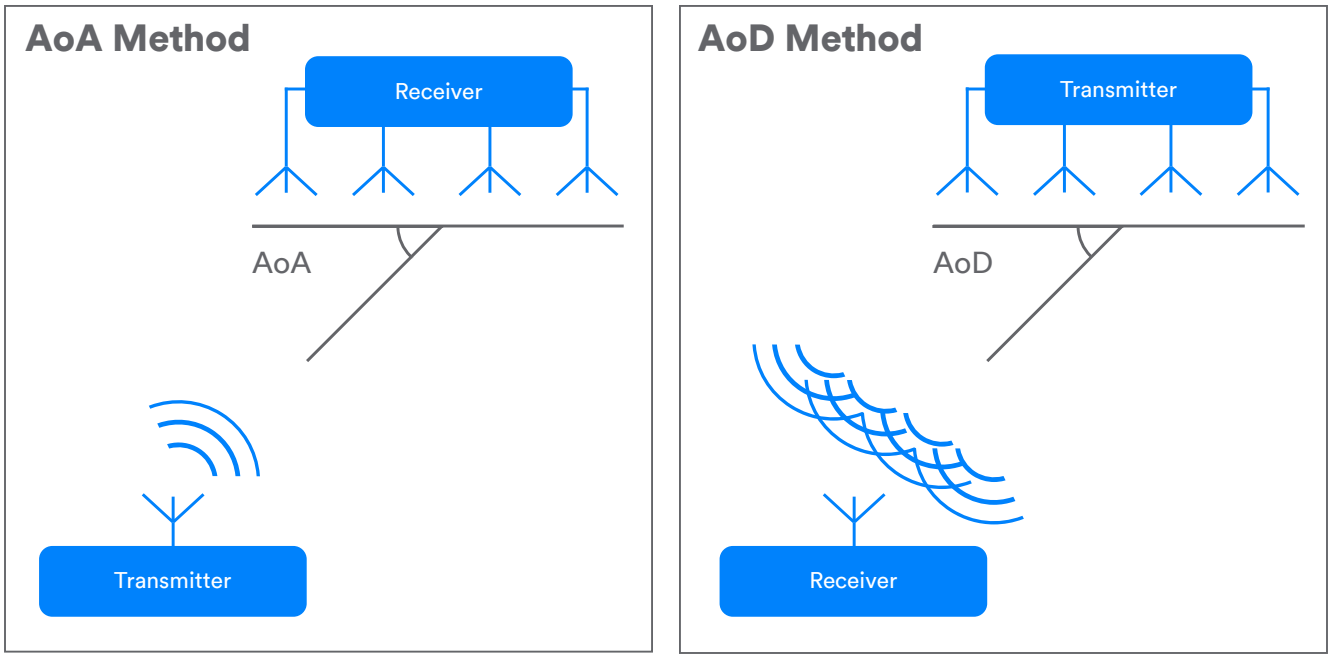


Figure 1 - Angle of Arrival (AoA) and Angle of Departure (AoD)

Bluetooth Core Specification v5.1 gives the Bluetooth Low Energy (LE) controller in the receiving device the ability to generate data that can then be used to calculate the directional angle to the transmitting device.

The addition of direction finding in this release of the Bluetooth Core Specification is the first of several steps in the Bluetooth roadmap that will ultimately enable key enhancements to Bluetooth location services. When the associated profiles have been released, Bluetooth developers will be able to exploit the new direction finding controller capability to create high accuracy, interoperable positioning systems such as real-time locating systems (RTLS) and indoor positioning systems (IPS).

The new direction finding feature also has the potential to enhance Bluetooth proximity solutions by determining device direction, particularly in directional item finding and point of interest information solutions.

Technical Details

The Bluetooth direction finding feature uses In-Phase and Quadrature (IQ) sampling to measure the phase of radio waves incident upon an antenna at a specific time. In the AoA approach, the sampling process is applied to each antenna in the array, one at a time, and in some suitable sequence depending on the design of the array.

Sampled data is passed up the stack via the Host Controller Interface (HCI) where it will then be possible to apply a suitable algorithm to the sampled data to calculate the direction of one device from the other. Algorithms for calculating angles from IQ samples are not defined in this core specification release. Once associated profiles are available, application developers will have the opportunity to implement algorithms suitable for the intended use case.

To support IQ sampling and the use of IQ samples by higher layers in the stack, the link layer (LL) and HCI have each changed.

At the link layer, a new field called the Constant Tone Extension (CTE) has been defined (see Figure 2). The purpose of the CTE field is to provide constant frequency and wavelength signal material against which IQ sampling can be performed. This field contains a sequence of 1s, is not subject to the usual whitening process and is not included in the CRC calculation.

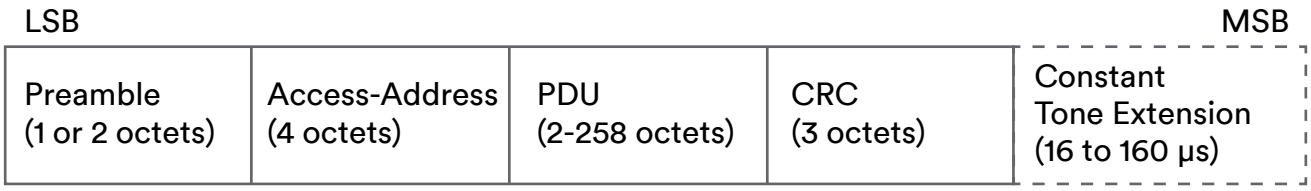


Figure 2 - Constant Tone Extension

CTE can be used in both connectionless and connection-oriented scenarios. For connectionless use, the periodic advertising feature is required (since deterministic timing in the sampling process is important) and CTE is appended to AUX_SYNC_IND PDUs. For connection-oriented use, new PDUs LL_CTE_REQ and LL_CTE_RSP have been defined. In either case, there are new HCI PDUs that allow the configuration of various aspects of CTE PDUs, such as the CTE length, length of the antenna switching pattern, and antenna IDs.

2.0 GATT Caching Enhancements

Background

All Bluetooth Low Energy connected devices use the Generic Attribute Profile (GATT). As such, the subject of GATT caching is of relevance to a wide range of device types.

GATT devices contain a database known as the *attribute table*. The attribute table contains GATT service, characteristic, and descriptor structural details and values, and is central to how GATT-based Bluetooth Low Energy devices work. Entries in the attribute table are identified by *attribute handles*.

GATT clients must perform a procedure known as *service discovery* to acquire details of the attribute table on the remote GATT server device that the client has connected to. The client can then use these details, including the identifying attribute handles in subsequent Attribute Protocol (ATT) interactions with the server.

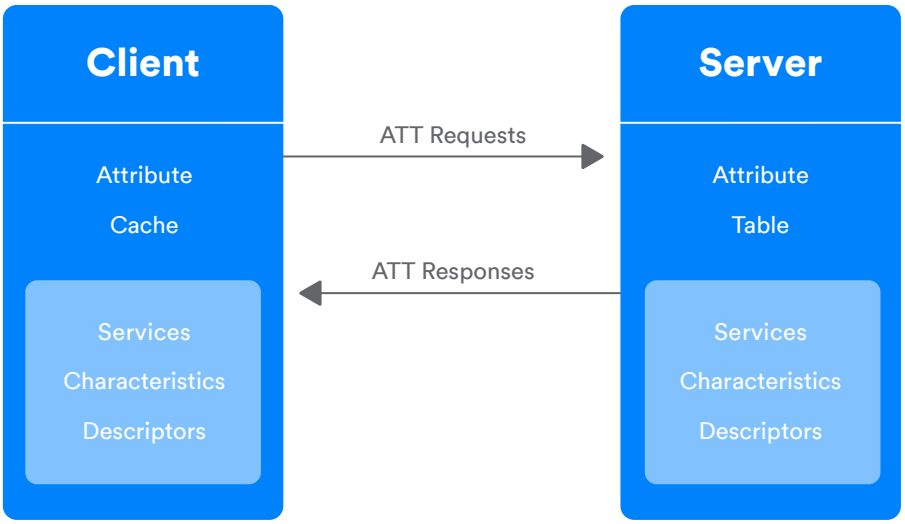


Figure 3 - Service Discovery and Attribute Caching

Some devices do not change their attribute table structure throughout their life. The GATT services, characteristics, and descriptors present in the table will always be the same, and only values of characteristics or descriptors will change. Other devices do change their attribute table from time to time.

Service discovery takes time and consumes energy. Therefore, Bluetooth Core Specification v5.1 defines an *attribute caching* strategy aimed at allowing clients to skip service discovery when nothing has changed.

Previously, caching and client/server attribute table synchronization was controlled solely using the *Service Changed* characteristic that might be present in the *Generic Attribute Service*. The GATT server could inform a connected client that its attribute table had changed by sending an ATT indication to the client. The client replied with an ATT confirmation and performed service discovery to synchronize its attribute cache with that of the server.

To avoid the GATT server needing to keep track of every client that ever connected to it, and whether or not each client had been informed of the latest attribute table change, previously the core

specification stipulated that clients and servers that have no trusted relationship (i.e. are not bonded) were required to perform service discovery every time they connect. This rule can cause energy efficiency and user experience issues for some types of products.

In addition, beyond making a single attempt to inform the client that the attribute table had changed using the ATT *Service Changed* indication, there was no further state management carried out with respect to the client's view of the attribute table vs the server's. The approach allowed a race condition in the communication between client and server, with respect to attribute table changes and general ATT interactions to exist, whereby it was possible for a client to time-out whilst waiting for a *Service Changed* indication after connecting to the server, proceed to send general ATT PDUs, and then receive a Service Changed indication.

Improved Caching Strategy

This release makes changes to how attribute caching and cache synchronization is approached by GATT clients and servers. It offers significant user-experience and energy-efficiency improvements by allowing clients without a trusted relationship with a server to retain their attribute cache across connections and resolves the race condition issue described above.

Two new characteristics, each a member of the *Generic Attribute Service,* have been introduced: *Database Hash* and *Client Supported Features*. Clients which do not have a trusted relationship with the server may now cache the attribute table across connections if the client supports the new Database Hash characteristic, as indicated by the client updating a flag in the server's *Client Supported Features* characteristic.

The *Database Hash* characteristic allows the client to ask the server if anything has changed, rather than relying on the server telling it using a Service Changed indication. The server is responsible for maintaining the value of the *Database Hash* characteristic, which is a hash value, calculated from pertinent aspects of the attribute table. The client reads its value immediately after establishing a connection. The client may cache the *Database Hash* value and subsequently use it to determine whether or not the remote attribute table has changed. If it has changed, the client performs service discovery again. If it has not, it does not need to. This offers a major user-experience and energy-efficiency benefit to some device types.

Furthermore, a client may now deduce that a device it is connecting to is the same type of device as one previously connected to and whose attribute table has already been cached by the client. If the database hash from the connected device is the same as the one associated with the client's attribute cache, and other details such as the device manufacturer are the same, the client may conclude that there is no need to perform service discovery for the connected device since the attribute cache obtained from another device contains equivalent data already.

For some applications, this change has considerable value. For example, consider Bluetooth smart locks, where a smartphone or other client device interacts with doors in a building to authenticate and open the door for a user when they approach. Service discovery need only be performed the first time the user attempts to pass through a door with a smart lock. The user may perceive a delay

in the door unlocking during this first occasion, but all subsequent times the user approaches any of the doors in building service discovery will not be required, and the user will experience a near instantaneous response from the smart lock.

Better State Management

A state machine defines whether or not the client view of the attribute table and the server view of its attribute table are in sync and, as such, whether or not the client needs to perform service discovery. The revised specification for attribute caching introduces the rigorously defined concept of *Robust Caching* that formalizes this state machine and introduces mechanisms for using it.

Clients are said to be in the *change-aware* state or are *change-unaware*. The specification lays out the precise rules for transitioning to the appropriate state and how to behave when in each of the two states.

Of particular note is the new «Database Out Of Sync» ATT error response that the server may return if it believes the client attribute table cache is out of sync with the server's. The server will ignore all ATT commands received from the client while it is in the change-unaware state. A number of events can transition the client's state to *change-aware*, including the server receiving an ATT confirmation to a *Service Changed* indication it had previously sent or the server having notified the client using the <<Database Out Of Sync>> error and subsequently receiving some other ATT PDU from the client. From the client's point of view, if it moves to the change-unaware state it will not use its attribute cache, regarding it as invalid. It will continue to be treated as invalid until the client's attribute cache and the servers are in sync once again.

3.0 Advertising Enhancement 1: Randomized Advertising Channel Indexing

Background

In Bluetooth Core Specification v5.0, advertising events are defined as "*one or more advertising PDUs sent on the primary advertising channel beginning with the first used advertising channel index and ending with the last used advertising channel index*".

In practice, this means that when all three channels are in use, as is often the case, advertising uses channels in the sequence 37 then 38 then 39, in strict order.

To lessen the possibility of persistent packet collisions, where two or more devices advertise on the same channel in an overlapping time period, Bluetooth Core Specification v5.0 stipulates that the time between consecutive advertising events must include a random delay of between 0 and 10ms.

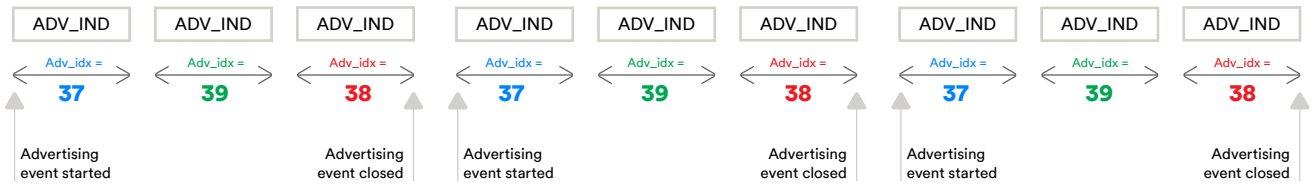


Figure 4 - Advertising channel use per the Bluetooth Core Specification 5.0 with the fixed sequence of 37, 38 then 39

Improved Packet Collision Avoidance

In this release, devices in the advertising state are no longer required to select advertising channels in a strict and unchanging sequence, starting with the lowest used channel index and ending with the highest. It is now permissible to select channel indices at random. The randomization of advertising channel indices further reduces the potential for advertising packet collisions occurring.

Applications that use advertising to perform connectionless communication will benefit from improved scalability and reliability in busy radio environments by implementing this change to advertising channel index selection.

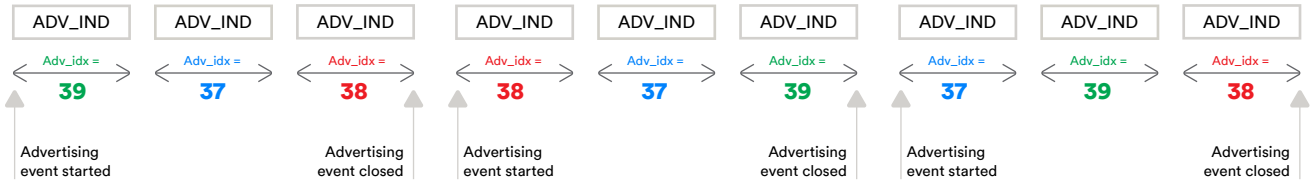


Figure 5 - Advertising channel use per the Bluetooth Core Specification 5.1 with a randomized channel index sequence

4.0 Advertising Enhancement 2: Periodic Advertising Sync Transfer

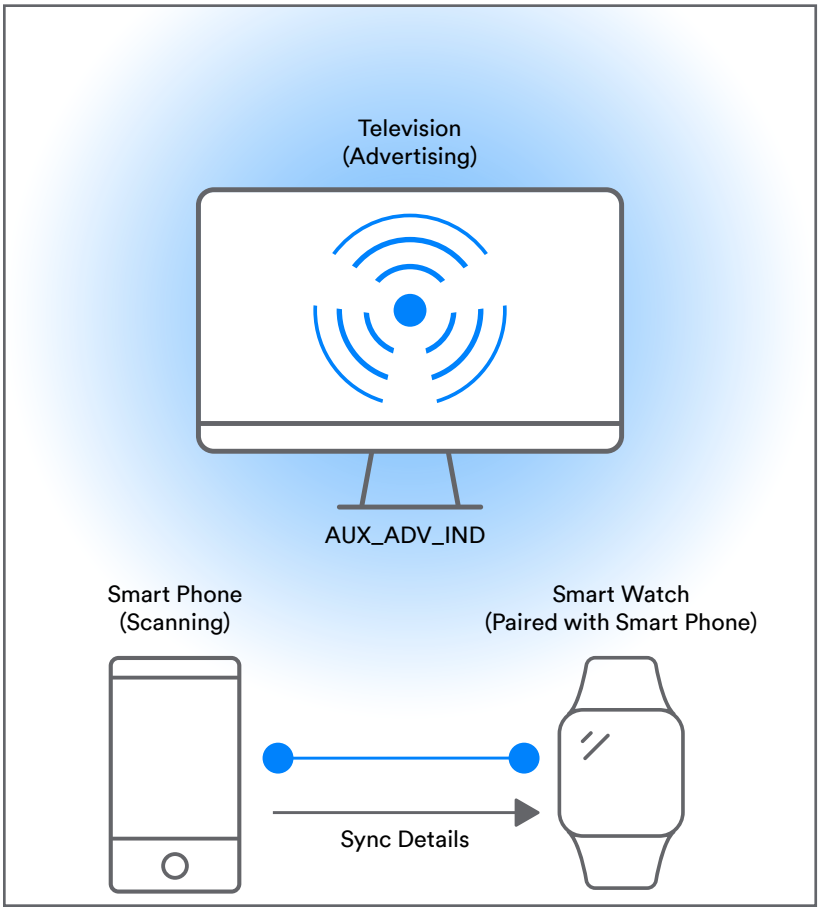
Background

Bluetooth Core Specification v5.0 introduced periodic advertising that uses deterministic scheduling of advertising events and provides a procedure that devices can use to synchronize their scanning with the advertising schedule of another device. Synchronization of the timing of scanning and advertising can make the scanning device more energy efficient and can make possible some use cases that require precise timing in the exchange of data.

To allow synchronization with the periodic advertising of a remote device, the remote device advertises AUX_ADV_IND PDUs which contain a field called SyncInfo. SyncInfo contains everything the receiving device needs to know to synchronize with the periodic advertising of AUX_SYNC_IND PDUs performed by the remote device from that point on. This periodic advertising synchronization procedure can be a relatively expensive operation, however.

The Power of Two

Some device types, with limited power, may not be able to afford the energy cost associated with the periodic advertising synchronization procedure or may have limitations in duty cycle or scan time that prevent it from working.



The new Periodic Advertising Sync Transfer (PAST) feature allows another, less constrained device to perform the synchronization procedure and then pass the acquired synchronization details over a point-to-point Bluetooth Low Energy connection to the other, constrained device. For example, a smartphone could scan for AUX_SYNC_IND packets from a TV and then pass them over a connection to an associated smart watch so that the watch can then benefit from using periodic advertising and scanning to acquire data from the TV.

Figure 6 - Periodic Advertising Sync Transfer usage example

Minor Enhancements

A number of minor enhancements are included in this release of the core specification.

HCI Support for Debug Keys in LE Secure Connections

Enhancement

LE secure connections is a Bluetooth pairing procedure that uses the Diffie Hellman key agreement protocol to secure the exchange of shared security keys during pairing. Diffie Hellman uses asymmetric, elliptic curve cryptography with a public and a private key. This makes it impossible to obtain the shared keys and use them for tracing and debugging connections during developing and testing.

In Bluetooth Core Specification v4.2, hard-coded key values for testing purposes were defined. But in cases where the Elliptic Curve algorithms are implemented in the controller, there was no way for the host to indicate it wanted to use them. The latest version of the core specification adds an HCI command that lets the host tell the controller to use the debug key values. Cases where the host implements the Elliptic Curve algorithms itself are not affected by this change.

Sleep Clock Accuracy Update Mechanism

Enhancement

Currently, when establishing an LE connection, the Central device informs the Peripheral how accurate its clock is using the Sleep Clock Accuracy (SCA) field. But the accuracy requirement might change depending on the concurrent use cases handled by the controller. For example, it might start at one value but need to be stepped up when another connection with higher clock-accuracy requirements is established.

Bluetooth Core Specification v5.1 provides a new link layer PDU, LL_CLOCK_ACCURACY_REQ, that can be used to inform connected Peripherals of new clock accuracy values. This PDU may be transmitted either by the Central to the Peripheral or by the Peripheral to the Central so that Peripherals may use it to inform Centrals in a connection of their clock accuracy.

This feature may result in lower power consumption in some cases.

ADI Field in Scan Response Data

Error Correction

The AdvDataInfo (ADI) field is used in extended advertising packets. Previously, this field was not allowed in scan response packets. In the latest core specification release it has become permissible to include ADI in scan response packets.

Interaction Between QoS and Flow Specification

Informative

This change is a clarification of the rules relating to Quality of Service (QoS) and Flow as they relate to Bluetooth Basic Rate/Enhanced Data Rate (BR/EDR).

Host Channel Classification for Secondary Advertising

Error Correction

The HCI command LE_Set_Host_Channel_Classification allows the classification of radio channels as “bad”. Previously its use applied only to connections, but now it applies to secondary advertising channels too.

Allow the SID to Appear in Scan Response Reports

Error Correction

The Advertising Set ID (SID) field is used in extended advertising packets. Previously this field was not allowed in scan response packets. In Bluetooth Core Specification v5.1 it has become permissible to include SID in scan response reports.

Specify the behavior when rules are violated

Informative

A new section, “Responding to Invalid behavior” has been added to the latest core specification release to clarify the rules which can be followed when dealing with a badly behaved Bluetooth device.