18. Bluetooth

Accessories that integrate Bluetooth technology must comply with the requirements stated in this chapter.

18.1 Conformity With Bluetooth Specifications

Every accessory that is compatible with an Apple product must support the *Bluetooth Core Specification* Version 2.1 + EDR or higher. This specification introduced the important security feature Secure Simple Pairing as well as Extended Inquiry Response.

18.1.1 Enhanced Data Rate

The Enhanced Data Rate (EDR) feature introduced in the *Bluetooth 2.0* specification enables accessories to communicate more efficiently. Every accessory must use EDR for the following reasons:

- It provides higher data rates compared to Basic Data Rate (BDR).
- It communicates more efficiently, transferring more data bits per unit of time.
- It reduces the power consumption used per bit transferred.
- It improves coexistence with Wi-Fi and other connected Bluetooth devices because it frees up more air time.
- It improves performance in multipoint configurations.

18.1.2 Adaptive Frequency Hopping

The Adaptive Frequency Hopping (AFH) feature introduced in the *Bluetooth 1.2* specification improves coexistence with Wi-Fi and other connected Bluetooth devices. Every accessory must use AFH.

18.1.3 Sniff Mode for Low Power Consumption

Minimizing power consumption is critical for all mobile devices. Therefore, every accessory that is compatible with an Apple product:

- Must support and must request Bluetooth sniff mode.
- Must accept requests for sniff mode and support all valid parameters listed in the Bluetooth specification.

- Must support a sniff interval of 15 ms.
- Should use the following recommended sniff mode values:
  - Max Interval: 15 ms
  - Min Interval: 15 ms
  - Sniff Attempt: 1
  - Sniff Timeout: 0
- Must not renegotiate sniff after being established.
- Must support sniff subrating.

Accessories that are compatible with Apple products must also use sniff mode as much as possible, especially when there is little or no data being transmitted over the Bluetooth link. Besides its power consumption advantages, sniff mode enables better antenna sharing with Wi-Fi.

The sniff mode parameters are specific to the usage model and Bluetooth Profile. The Apple product expects the accessory to request sniff mode with appropriate parameters for a specific usage. If the accessory does not send such a request, the Apple product may send a sniff mode request. When the Apple product sends a request for sniff mode, the remote device must accept the request and its parameters without negotiation.

If the accessory sets the sniff mode parameters, the accessory must set the sniff interval to less than a third of the Bluetooth baseband Link Supervision Timeout. This makes the Bluetooth link less susceptible to interference. To improve link robustness, the accessory must use a shorter sniff interval instead of multiple sniff attempts.

Links with a sniff interval of 1 second or more make the slave device open up a large correlation window, which has to be taken into account when calculating the number of sniff attempts. With sniff intervals shorter than 1 second, multiple sniff attempts can improve link robustness but will increase power consumption.

18.1.4 Role and Topology Management

Every accessory that is compatible with an Apple product must:

- Accept a request for Role Switch from an Apple product.
- Continue with the connection when the Apple product rejects a request for Role Switch.

In a Bluetooth connection, one device is the master and the other the slave. The master can have multiple slaves, thus forming a piconet. The master device can also be a slave role to another master, creating a scatternet.

Such a scenario creates complications since the device has to alternate between the two piconets and thus wastes valuable bandwidth. Managing the topology of the network is therefore important for maximum performance. The Apple product may request a Role Switch, depending on its current topology, and the remote device must accept the request. The Apple product may also reject a request for a Role Switch because of topology concerns. Having a suboptimal topology may degrade the audio quality and the user's experience.

Only when it is maintaining multiple links, either Bluetooth or Wi-Fi, will the Apple product request or deny role switches. Hence, it will grant a role switch if there is no reason for the Apple product to be master. It is expected that the accessory will behave the same, only trying to be master when there is a legitimate reason.

The accessory must not always request to be master by default if there is no need in the system topology to do so. If later the accessory needs to be master in order to maintain multiple links, it must ask to be master at that time.

18.1.5 Extended Inquiry Response

Every accessory that is compatible with an Apple product must provide the following information in its Extended Inquiry Response packet:

- The Local Name of the accessory (Complete or Shortened).
- The TX Power Level.
- The Service Class UUID for the iAP2 protocol, if applicable (see iAP2 (page 357)).

During the Bluetooth discovery process, the Apple product prefers to display the Friendly Name of discovered accessories. Before the 2.1 version of the Bluetooth specification the Apple product would have to set up a connection to the accessory and do a Remote Name Request, which takes power, antenna time, and user's time. The Extended Inquiry Response feature, introduced in Bluetooth 2.1, lets an accessory send its Local Name and other information as part of the Inquiry Response and thereby increase the speed and efficiency of the discovery process.

The Local Name must match the accessory's markings and packaging and not contain ':' or ';'.

18.1.6 Secure Simple Pairing

Every accessory that is compatible with an Apple product must:

- Use Secure Simple Pairing.
- Use the Numerical Comparison method if it has a display and input device supporting it.

Secure Simple Pairing greatly increases security and is a mandatory security feature introduced in the Bluetooth 2.1 specification. To protect against a man-in-the-middle attack, the Numerical Comparison association model must be used whenever feasible. See Volume 1, Section 5.4 in the *Bluetooth Core Specification* , Version 2.1 + EDR.

18.2 Profiles

The Apple knowledge base article support.apple.com/kb/ht3647 provides a complete list of the Bluetooth profiles that certain Apple devices support. The Bluetooth specifications are the starting point for designing accessories that are compatible with these products. The following sections add information and requirements for some profiles, which can help accessory developers achieve superior results.

18.2.1 Device ID Profile (DID)

Every accessory that is compatible with an Apple product must:

- Support the Bluetooth Device ID Profile, version 1.3 or higher.
- Use the company identifier from the Assigned Numbers specification assigned by the Bluetooth SIG as its Vendor ID value (VID). See http://www.bluetooth.org/Technical/AssignedNumbers/identifiers.htm (requires login). Bluetooth HID Profile accessories may use a VID assigned by the USB Implementers Forum (USB-IF at http://www.usb.org) if the manufacturer does not have a Bluetooth SIG company identifier.
- Use its VID value for the end product manufacturer.
- Use the Vendor ID Source field to identify which organization assigned the value used in Vendor ID field value. See Section 5.6 of the *Bluetooth Device ID Profile Specification* .
- Use a ProductID value that uniquely identities the product.
- Use a Version value that uniquely identifies the software version. If the accessory supports iAP2, then this value must match the Firmware Version used in the Accessory Identification message (see Accessory Identification (page 265)).

The Device ID profile lets the Apple product identify the implementation of the remote accessory. This is valuable information and can be used to bridge alternate interpretations of the Bluetooth specification when communicating with a remote accessory. Therefore it is important that the information in the Device ID record uniquely identify the implementation.

In the case of Bluetooth car kit devices, for instance, the same car kit might go into two different car models. Ideally the two car kits must have different Product IDs. However, it is acceptable for them to have the same ProductID as long as they have identical hardware, software, and features. If the implementations differ at all, they must have different Product IDs. The accessory can also use a secondary Device ID to uniquely identify the product ID or model number.

18.2.2 Hands-Free Profile (HFP)

Every accessory that is compatible with an Apple product and supports the Handsfree Profile must meet the requirements of the Bluetooth *Hands-Free Profile* specification, Version 1.5 or higher. Additional Apple requirements are specified in this section.

Remote accessories can use the Bluetooth *Hands-Free Profile* for phone communications. To achieve the best user experience, the remote accessory must support the following features, which are optional in the Bluetooth specification.

18.2.2.1 Remote Audio Volume Control

Every accessory that is compatible with an Apple product and supports HFP must:

- Support Remote Audio Volume Control so the speaker volume on the Hands-Free accessory can be controlled from the Apple product as described in Section 4.28 in the Bluetooth *Hands-Free Profile* specification version 1.5.
- Set the Remote volume control bit in the Supported Features bitmap sent with the `AT+BRSF=` command.

In some situations it is easier for the user to control the output volume through the Apple product instead of directly on the remote accessory. For example, a passenger (or-if the car is parked-the driver) in a car could use the volume slider on the phone to control the audio volume. Volume control synchronization is outlined in Section 4.48.2 in the Bluetooth *Hands-Free Profile* specification version 1.5.

18.2.2.2 Indicator Event Reporting

Every accessory that is compatible with an Apple product and supports HFP must use indicator events reporting and not perform repetitive polling of status.

Apple products support all mandatory and optional indicators specified in HFP version 1.5 (service, call, callsetup, callheld, signal, roam, battchg). To minimize unnecessary polling of status using the `AT+CIND?` command, the remote accessory must enable indicator events reporting by sending an `AT+CMER` command. The Apple product will then send a `+CIEV` event when there is a change in status of an indicator. The remote accessory must request the initial status using the `AT+CIND=?` and `AT+CIND?` commands, according to the HFP specification.

18.2.2.3 Voice Recognition Activation

Every accessory that is compatible with an Apple product and supports HFP must:

- Support Voice Recognition Activation, both AG and HF initiated as described in Section 4.25 in the Bluetooth *Hands-Free Profile* specification version 1.5.
- Set the "Voice recognition activation" bit in the "SupportedFeatures" bitmap sent with the `AT+BRSF=` command.

Apple products support voice recognition initiated by remote (Hands-Free) accessories and iOS (Audio Gateway) accessories.

18.2.2.4 Echo Cancellation and Noise Reduction

When echo cancellation and noise reduction are performed locally on a Hands-Free accessory, it must turn off echo cancellation and noise reduction on the Apple product by sending an `AT+NREC` command, as described in Section 4.24 in the Bluetooth *Hands-Free Profile* specification version 1.5.

Apple products support echo cancellation and noise reduction; these features are active by default. If a Hands-Free accessory also does echo cancellation and noise reduction it needs to turn these features off on the Apple product (the Audio Gateway). This avoids unnecessary degradation of audio quality due to double audio processing.

18.2.2.5 In-Band Ringing

Every accessory that is compatible with an Apple product and supports HFP must also support In-Band Ringing as specified in Section 4.13.1 in the Bluetooth *Hands-Free Profile* specification version 1.5. If the user sets a ring tone on the Apple product, the same ring tone must sound on the hands-free accessory.

18.2.2.6 Synchronous Connection

Every accessory that is compatible with an Apple product and supports HFP must:

- Support eSCO parameter set S2 and S3 and accept requests for these settings. See Section 5.6 of the Bluetooth *Hands-Free Profile* specification version 1.5.
- Request eSCO parameter set S2 or S3 when setting up a Synchronous Connection. Note that eSCO parameter set S1 must not be requested.
- Render audio within 40 ms after the SCO/eSCO connection has been set up.

The eSCO packet types offers retransmission of packets; traditional SCO packets are not retransmitted. This improves audio quality and the user's experience. The eSCO packet types 2-EV3 and 3-EV3 offer a greater time interval between packets, which can improve Wi-Fi performance and allow time for other concurrent Bluetooth connections to send data. Apple strongly recommends the use of 2-EV3 and 3-EV3 packets for SCO connections. Using HV3 packets is highly discouraged. HV3 packets require more link time and does not allow for retransmission of audio packets which impacts the audio performance in presence of RF interference.

18.2.2.7 Wide Band Speech

Every accessory that is compatible with an Apple product and supports HFP must support a Wide Band Speech Connection as described in Section 5.7.4 of the Bluetooth *Hands-Free Profile* specification version 1.6. If Wide Band Speech Connection is supported, it must support the T2 link parameter settings.

All Apple devices running iOS 5 or later support Wide Band Speech. If both the Apple device and the accessory support Wide Band Speech then Wide Band Speech link will be used for eSCO connection for use cases like cellular calls, FaceTime and Siri.

18.2.3 Message Access Profile (MAP)

Every accessory that is compatible with an Apple product and supports MAP must:

- Support Message Notification as described in Section 4.1 of the Bluetooth *Message Access Profile* specification, version 1.0.
- Register for notifications immediately after the connection is established, as described in Section 4.5 in the *Message Access Profile* specification, version 1.0.
- Not expect the TEL property to be present in the originator VCARD (the properties N and FN will be included). See Section 3.1.3 in the *Message Access Profile* specification, version 1.0.
- Not provide a user interface for sending messages. Apple devices do not support sending messages using MAP.

All Apple devices running iOS 6.0 or later support MAP.

18.2.4 Audio/Video Remote Control Profile (AVRCP)

Every accessory that is compatible with an Apple product and supports the Audio/Video Remote Control Profile must meet the requirements of the Bluetooth *Audio/Video Remote Control Profile* specification, Version 1.4. Additional Apple requirements are specified in this section.

18.2.4.1 Supported Operations

Apple products support the following operation_IDs in Pass Through commands:

- Play
- Stop

- Pause
- Fast Forward
- Rewind
- Forward
- Backward

18.2.4.2 Repeat and Shuffle Modes

Every Apple device supports Repeat and Shuffle modes in the role of an AVRCP target. An AVRCP controller may use `SetPlayerApplicationSettingValue` to set a value on the Apple device and `GetPlayerApplicationSettingValue` to read a value, as described in Sections 6.5.4 and 6.4.3 of the Bluetooth *Audio/Video Remote Control Profile* specification version 1.4.

18.2.4.3 Notifications

Every accessory that is compatible with an Apple product and supports AVRCP must register for notifications and not perform repetitive polling to determine the status of the Apple product.

Every Apple device supports registering for notifications in the role of an AVRCP Target, as described in Section 6.7 of the Bluetooth *Audio/Video Remote Control Profile* specification version 1.4. The commands `RegisterNotification` and `GetPlayStatus` are supported for these notifications:

- EVENT_PLAYBACK_STATUS_CHANGED
- EVENT_TRACK_CHANGED
- EVENT_NOW_PLAYING_CONTENT_CHANGED
- EVENT_AVAILABLE_PLAYERS_CHANGED
- EVENT_ADDRESSED_PLAYER_CHANGED
- EVENT_VOLUME_CHANGED

Accessories must use AVRCP notifications to present Apple device state to the user in accordance with the requirements in Presentation of Apple Device Updates (page 61).

18.2.4.4 Play/Pause Button

All accessories that support AVRCP and implement a Play/Pause button must confirm the playback status of the Apple device via AVRCP notifications (see Notifications (page 351) before sending a Play or Pause command (see Supported Operations (page 350)). Specifically:

- If the Apple device has notified the accessory that it is paused, pressing the accessory's Play/Pause button must send a Play command.

- If the Apple device has notified the accessory that it is playing, pressing the accessory's Play/Pause button must send a Pause command.
- The accessory must not infer Apple device playback status based on the number of times the Play/Pause button has been pressed.

18.2.4.5 Volume Handling

Every accessory that is compatible with an Apple product and supports AVRCP must support Absolute Volume, as described in Section 6.13 of the Bluetooth *Audio/Video Remote Control Profile* specification version 1.4.

Every Apple device supports volume handling in the role of AVRCP Controller.

18.2.4.6 Browsing

Every accessory that is compatible with an Apple product and supports Browsing (in controller role) as part of AVRCP must:

- Not try to index or cache the entire library upon connection. The Apple product may contain tens of thousands of media items, each present multiple times in the hierarchy.
- When browsing a specific folder, do not fetch all its items. Only fetch those that are displayed to the user. It may prefetch a few items to improve the responsiveness of the user interface.
- Not reorder items (e.g. alphabetically).
- Not assume UIDs to be statically defined, especially in the root folder. The ordering and UIDs of folders and items may change at any point in future releases.
- Send the `SetBrowsedPlayer` command after receiving an EVENT_UIDS_CHANGED notification.
- Not assume that the UID passed to the PlayItem command will result in the media player playing that UID.

Currently only the built-in Music app supports browsing. When switching between players, an EVENT_AVAILABLE_PLAYERS_CHANGED notification and an EVENT_ADDRESSED_PLAYER_CHANGED notification will be generated. The UI then needs to look at the feature bit mask of the listed player to determine whether browsing is currently available.

All Apple devices running iOS 6.0 or later support AVRCP Browsing.

18.2.4.7 iOS App-Provided Metadata

An audio app running on an Apple device may use the iOS MediaPlayer Framework APIs to provide metadata about the current audio stream. The Apple device supplies this metadata to the accessory using AVRCP. For more information, see the `MPNowPlayingInfoCenter` class in Apple's MediaPlayer Framework documentation.

18.2.5 Advanced Audio Distribution Profile (A2DP)

Every accessory that is compatible with an Apple product and supports the Advanced Audio Distribution Profile must meet the requirements of the Bluetooth *Advanced Audio Distribution Profile* specification, Version 1.2. Additional Apple requirements are specified in this section.

18.2.5.1 SubBand Codec (SBC)

The SBC Codec Specific Information Elements, defined in Section 4.3.2 of the A2DP specification, that are applicable to Apple products are listed in Table 18-1 (page 353).

Table 18-1 SubBand Codec Information Elements for Apple products

Element	Value
Sampling Frequency	44,100 Hz
Channel Mode	Stereo
Block Length	16
Subbands	8
Allocation Method	Loudness
Bitpool range	2 to 53. Accessories for Apple products must support 53.

18.2.5.2 MPEG 2/4 AAC Codecs

Apple devices support the non-mandatory codec MPEG-2/4 AAC, as defined in Section 4.5 of the *Advanced Audio Distribution Profile* specification, Version 1.2. Accessories must use the AAC codec in addition to SBC, because it provides higher audio quality for a given bit rate.

Note: The following specifications provide details of Apple's implementation of the MPEG-2/4 AAC codec. In case of conflicts, the A2DP specification governs.

The MPEG 2/4 AAC Codec Specific Information Elements, defined in Section 4.5 of the A2DP specification, that are applicable to Apple devices are listed in Table 18-2 (page 353).

Table 18-2 MPEG-2/4 AAC Codec Information Elements for Apple devices

Element	Value
Object Type	MPEG-2 AAC LC

Element	Value
Sampling Frequency	44,100 Hz
Channels	2
Bit rate	264,630 bps
VBR	0

AAC audio stream packets in Apple devices have the structure shown in Table 18-3 (page 354).

Table 18-3 AAC audio packet for Apple devices

L2CAP	AVDTP	MPEG-4 LATM	MPEG-4 AAC
Header	Header	AudioMuxElement	Audio Payload

The AAC Media Payload Format, as defined in Section 4.5.4 of the A2DP specification, is formatted using LATM, as defined in Section 4 of *IETF RFC 3016* . The following notes apply to the packet fields shown in Table 18-3 (page 354).

- The suggested L2CAP MTU value for each Apple device's AAC streaming channel is 885 bytes.
- The AVDTP Header is shown as the RTP header in Figure 4 of RFC 3016, and is the header defined in Section 7.2.1 of the Bluetooth *Audio/Video Distribution Transport Protocol* , Version 1.2.
- The `AudioMuxElement` is the same as the RTP payload in RFC 3016. It is defined in Section 1.7.3, Table 1.32 in ISO/IEC 13818-3:2005, subpart 1. The `muxConfigPresent` argument to the `AudioMuxElement` is set to 1 (in-band mode), as recommended in Section 4.1 of RFC 3016. As recommended in Section 4.3 of RFC 3016, only one `AudioMuxElement` is put into each AVDTP packet.
- The audio payload is encoded using MPEG-4, as recommended in Section 4.5.4 of the A2DP specification.
- For AAC-LC support, the accessory must support VBR capability. The Apple device will be varying AAC bit rate depending on the content and the accessory must be able to handle the variation without causing gap in the audio.

18.3 Audio Routing

This section describes how an accessory can differentiate between various audio contents coming from an Apple device and use this information to decide playback behavior.

An accessory can receive audio data from the Apple device via either of two Bluetooth profiles:

- HFP using eSCO channel
- A2DP using ACL channel

The Apple device picks which channel to use depending on how the audio content is used. An audio path created for two way communication (such as phone calls or FaceTime) always uses the HFP (eSCO) route for sending audio data. Music and similar content uses the A2DP route. In the absence of a defined route, audio playback will default to the Apple device.

If the accessory is already connected to the Apple device using the Lightning or 30-pin connector, connecting using Bluetooth must not pause any playing audio.

18.3.1 Audio Data Received via HFP Profile

Most of the audio content sent via HFP (eSCO) routes requires two way communication. Cases where HFP (eSCO) is used include (but are not limited to) cellular calls, FaceTime, and voice mail.

For any audio content that is being received via the HFP (eSCO) route, it is expected that both the speaker and the microphone of the accessory are dedicated to the Bluetooth link and must not handle any other audio content.

18.3.2 Audio Data Received via A2DP Profile

Audio content transferred via A2DP profiles can be broadly classified into two categories:

- Audio content from music, video, or game-like applications.
- System-generated sound for alerts and notifications.

18.3.2.1 Differentiating Audio Content from System Sounds

Music-like content can be differentiated from system sound by adding support for Audio/Video Remote Control Profile (AVRCP) version 1.3 or later. The AVRCP profile allows an accessory to be aware of the audio playback state in the Apple device, using notifications. See Audio/Video Remote Control Profile (AVRCP) (page 350).

When an Apple device initiates audio playback over an A2DP channel for playing music content, an AVRCP notification EVENT_PLAYBACK_STATUS_CHANGED is sent to indicate that playback status has changed to play state. See Section 6.7.2 of the *Audio/Video Remote Control Profile* specification, version 1.4. This indicates that audio data via the A2DP profile contains music. When an Apple device initiates audio playback over an A2DP channel for playing system sound, no AVRCP notification is sent.

Figure 18-1 (page 356) and Figure 18-2 (page 356) show the difference between the notifications for music playback and for system sounds.

Figure 18-1 Initiate Audio Playback (e.g. music)

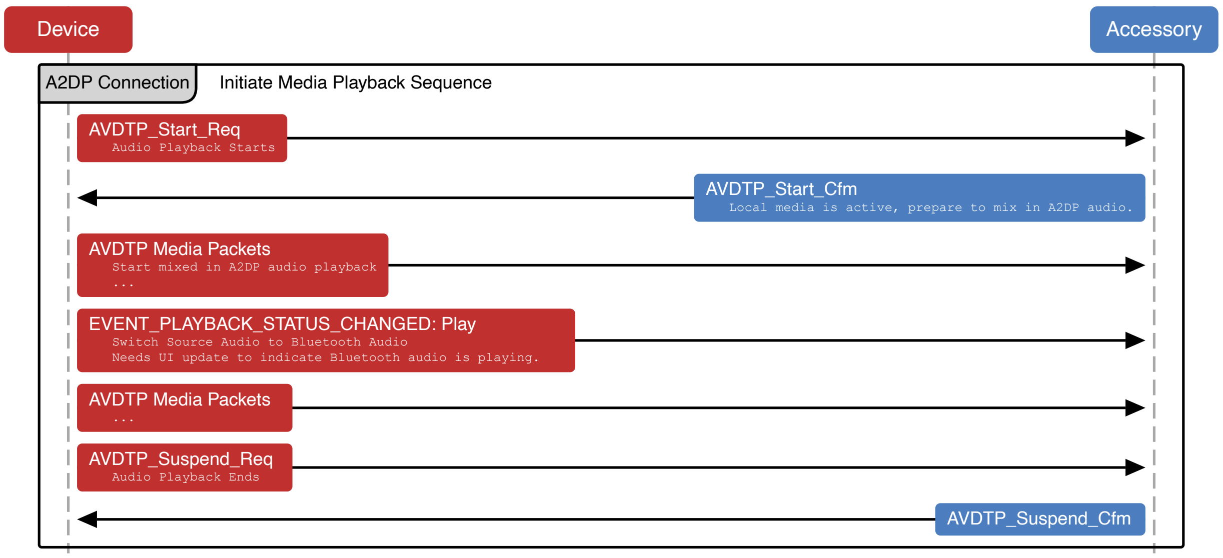
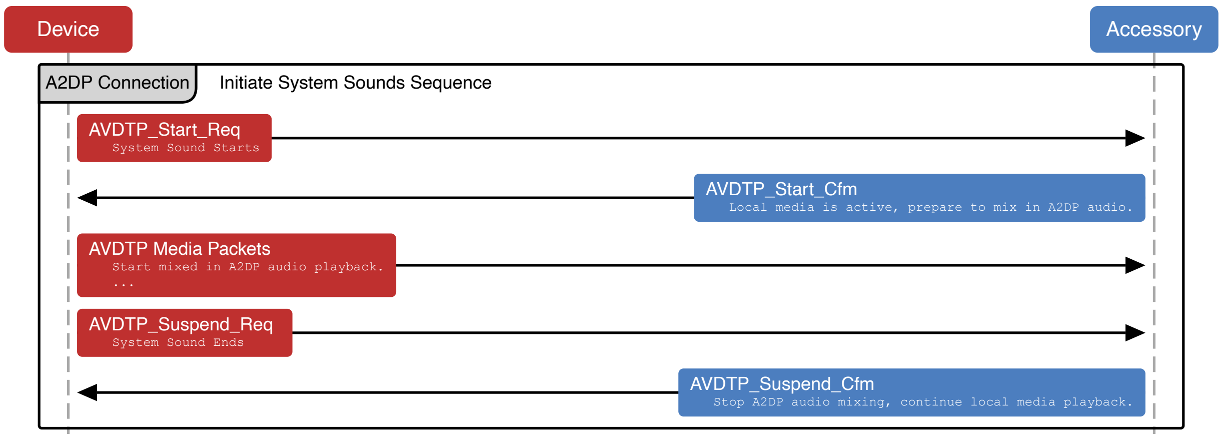

Device
Accessory
A2DP Connection
Initiate Media Playback Sequence
AVDTP_Start_Req
Audio Playback Starts
AVDTP_Start_Cfm
Local media is active, prepare to mix in A2DP audio.
AVDTP Media Packets
Start mixed in A2DP audio playback
...
EVENT_PLAYBACK_STATUS_CHANGED: Play
Switch Source Audio to Bluetooth Audio
Needs UI update to indicate Bluetooth audio is playing.
AVDTP Media Packets
...
AVDTP_Suspend_Req
Audio Playback Ends
AVDTP_Suspend_Cfm

Figure 18-2 Initiate System Sound (e.g. turn-by-turn directions)


Device
Accessory
A2DP Connection
Initiate System Sounds Sequence
AVDTP_Start_Req
System Sound Starts
AVDTP_Start_Cfm
Local media is active, prepare to mix in A2DP audio.
AVDTP Media Packets
Start mixed in A2DP audio playback.
...
AVDTP_Suspend_Req
System Sound Ends
AVDTP_Suspend_Cfm
Stop A2DP audio mixing, continue local media playback.

18.3.2.2 Expected Audio Routing Behavior for A2DP

The accessory must tune its audio routing behavior based on audio content over A2DP channel.

If audio data contains music, then it is expected that the accessory speakers are dedicated to audio data coming via the Bluetooth link and any other audio playback is paused. If audio data contains system sound, then it is expected that the accessory can render audio as desired. If the accessory is playing audio from a different source, then system sound data can be mixed with the existing track for playback; it is not necessary to pause existing audio playback on the device.

18.4 iAP2

Accessories that implement iAP2 over the Bluetooth transport must additionally meet the following requirements:

- The Bluetooth Service Discovery Protocol (SDP) must be supported.
- The SDP data Maximum Transmission Unit (MTU) must be at least 672 bytes.
- SDP records must not be fragmented.
- Extended Inquiry Response (EIR) must be supported.
- A Service Class UUID of `0xFFCACADEAFDECADEDEFACADE00000000` must be declared in both SDP and EIR.
- The EIR Local Name must be the same as the `Name` parameter in the IdentificationInformation (page 807) message.

Unlike iAP1, there is no requirement for a specific Class of Device (CoD) or Major Service.

Apple recommends the following for a successful Bluetooth connection:

- Implement Bluetooth Sniff Mode, or Sniff Subrating if Bluetooth 2.1 is being used.
- Let the Apple device be the master device.
- For large packet transfers, stay within a Maximum Transmission Unit (MTU) size of 1000 bytes and a minimum of 658 bytes at the iAP2 layer.

Unlike wired transports, Apple devices will enter a hibernate state regardless of how many active Bluetooth connections are present. Bluetooth traffic from the accessory to the device will cause the device to exit the hibernate state. Therefore, accessories must generate Bluetooth traffic to an Apple device only in response to a direct user action. Autonomous generation of Bluetooth traffic for the purpose of keeping an Apple device out of hibernate is grounds for failure to pass self certification.

To re-establish a Bluetooth connection to the Apple device, the accessory must make a Bluetooth SDP query to find the RFCOMM channel associated with the Service Class UUID of `0xFECACADEAFDECADEDEFACADE00000000`, then connect to that channel. The accessory must not assume that the channel will remain the same between connections.

The Apple device may refuse a Bluetooth connection under these circumstances:

- There are too many Bluetooth connections made to the Apple device. In this case, the accessory will receive an error indicating that a resource is unavailable.
- There are too many RFCOMM protocol connections to the Apple device. In this case, the accessory will receive a Resource Denied error.

The accessory must not expect that the Apple device will try to re-establish a broken Bluetooth connection.

Note: Service Class UUID and RFCOMM UUID are listed in little-endian order.

18.5 Test Procedures

18.5.1 Pairing and Connection Establishment

18.5.1.1 Apple Device to Accessory

1. Search for accessory from Apple device and initiate paring.
2. Pair should be successful after exchanging pin code.

18.5.1.2 Car Kit to Apple Device

1. Search for Apple device from accessory and initiate paring (Apple device needs to be in Bluetooth menu).
2. Pair should be successful after exchanging pin code.

18.5.1.3 Sniff Mode

1. Leave the connection idle for 30 seconds.
2. Device should enter sniff mode upon request.

18.5.2 Reconnection

18.5.2.1 Reconnect from Apple Device

1. Toggle Airplane Mode and reconnect BT from Apple device.
2. Reconnection should be successful.

18.5.2.2 Reconnect from Accessory

1. Toggle Airplane Mode and reconnect BT from accessory.
2. Reconnection should be successful.

18.5.2.3 Reconnect During Active Call

1. Power cycle accessory during active call.
2. Reconnection should be successful. Audio should be available in uplink and downlink.

18.5.2.4 Range

1. Take Apple device out of range during active call. Return to BT range after 2 minutes and initiate BT connection from Apple device/accessory.
2. Audio should be routed to Apple device after disconnection. Audio should be available in uplink and downlink after reconnection.

18.5.3 Indicators

1. Check the Human Machine Interface (HMI) after BT connection.
2. Support indicators should display correct values for: Battery, Signal Strength, Roaming Indicator, and Carrier Name.

18.5.4 Outgoing Call

18.5.4.1 Dialed from Accessory

1. Enter phone number on accessory's HMI.
2. Dialing tone should be heard over SCO/eSCO link before call is answered by the remote party. Audio should be available in uplink and downlink once call is answered.

18.5.4.2 Cancel Call from Accessory

1. Enter phone number on accessory's HMI. Wait for dialing to start and press End Call.
2. Dialing tone should be heard over SCO/eSCO link.

18.5.4.3 Dialed from Apple Device

1. Enter phone number on Apple device.
2. Dialing tone should be heard over SCO/eSCO link before call is answered by the remote party. Audio should be available in uplink and downlink once call is answered.

18.5.4.4 Dialed from Downloaded Call History

1. On accessory, dial to an entry in downloaded call history.

2. Dialing tone should be heard over SCO/eSCO link before call is answered by the remote party. Audio should be available in uplink and downlink once call is answered.

18.5.4.5 Audio Transfer

1. From accessory/Apple device, transfer audio between Apple device and accessory.
2. Audio should be routed to the correct source after each action.

18.5.4.6 Audio Quality

1. Maintain a call for 10 minutes and check audio quality.
2. Audio quality should be decent in uplink and downlink.

18.5.5 Incoming Call

18.5.5.1 In-Band Ringtone

1. Make incoming call and check audio quality for in-band ringtone.
2. Ring tone should be complete and should sound clear.

18.5.5.2 Caller ID

1. Make incoming call and check caller ID on accessory's HMI.
2. Caller ID should be displayed in a good format.

18.5.5.3 Call Rejection

1. Make incoming call and reject from accessory/Apple device.
2. Call should be rejected.

18.5.5.4 Answer Call from Accessory

1. Make incoming call and answer from accessory.
2. Audio should be available in uplink and downlink once call is answered.

18.5.5.5 Answer Call from Apple Device

1. Make incoming call and answer from Apple device.
2. Audio should remain on Apple device (Privacy mode).

18.5.5.6 Audio Transfer

1. From accessory/Apple device, transfer audio between Apple device and accessory.
2. Audio should be routed to the correct source after each action.

18.5.5.7 Audio Quality

1. Maintain a call for 10 minutes and check audio quality.
2. Audio quality should be decent in uplink and downlink.

18.5.6 FaceTime Audio

18.5.6.1 Incoming FaceTime Audio Call

1. Use the same steps as Incoming Call (page 360).

18.5.6.2 Outgoing FaceTime Audio Call

1. Use the same steps as Outgoing Call (page 359).

18.5.7 Three Way Call

18.5.7.1 Single Call Hold/Resume

1. During a call, press Hold / Resume on HMI.
2. Call should be held / resumed after each action.

18.5.7.2 Ignore Second Incoming Call

1. During an active call, press Reject on HMI to ignore the second incoming call.
2. Second call should be rejected. First call should remain unchanged.

18.5.7.3 Replace Call

1. During an active call, press Replace to end current call and answer the second call.
2. Second call should be answered. First call should be ended.

18.5.7.4 Answer Second Call

1. During an active call, press Answer to the second incoming call.

2. Second call should be answered. First call should be put on hold.

18.5.7.5 Call Swap

1. Swap between active and held calls from HMI.
2. Call status should change in response to each action.

18.5.7.6 Conference

1. Join 2 calls from HMI. Make sure Hold/Resume/End functions work for conference calls.
2. Each party should be able to hear audio from the other two. Hold/Resume/End functions should work for conference calls.

18.5.8 Enhanced Call Control

18.5.8.1 Specify a Call to End in a Three Way Call

1. Form a conference call and specify to end one of the calls.
2. The specified call should end.

18.5.8.2 Specify a Call to Hold in a Three Way Call

1. Form a conference call and specify to put on call on hold (split the conference).
2. The specified call should become held from active.

18.5.9 Visual Voice Mail

18.5.9.1 VVM Playback

1. Start VVM playback.
2. Virtual call should be displayed by HMI. Audio should be clear.

18.5.9.2 VVM Playback During A2DP

1. Start VVM playback during A2DP streaming.
2. Virtual call should start and should remain undisrupted. Audio should be clear.

18.5.9.3 Call Back During VVM Playback

1. Play VVM and press Call Back on Apple device.
2. Transition from VVM to outgoing call should be smooth.

18.5.9.4 Incoming Call During VVM Playback

1. Play VVM and make an incoming call.
2. Transition from VVM to incoming call should be smooth.

18.5.10 Siri

1. Trigger Siri from either Apple device or accessory.
2. Beeps should be complete and sound clear.
3. Ask Siri a question (time, weather, etc.).
4. Audio should be complete and sound clear. Virtual call should end soon after Siri provides the answer.
5. Ask Siri to call someone.
6. Transition from Siri to outgoing call should be smooth.
7. Ask Siri to play certain tracks.
8. Siri should find the right track to play.
9. Ask Siri for directions (Starbucks, Whole Foods, etc).
10. Siri should find the directions and start turn-by-turn navigation.

18.5.11 A2DP

18.5.11.1 Audio Quality

Assess the quality of the audio signal in each of the following scenarios:

1. Stream music from Music app.
2. Stream music using iTunes Radio.
3. Stream audio using Podcast app.
4. Stream audio using Beats Music app.
5. Stream audio using 3rd party apps (Pandora, Spotify, etc).

18.5.11.2 Audio Switching

1. During A2DP streaming, switch audio back to Apple device and switch back to accessory.
2. Audio should be routed to the intended source. Audio quality should be good switching back to BT.

18.5.11.3 HFP Interaction

1. Make incoming / outgoing call during A2DP.
2. Audio should be suspended during the call and resume after the call.

18.5.11.4 Siri

1. Trigger Siri during A2DP.
2. Audio should be resumed after the Siri session.

18.5.11.5 Video Playback

1. Stream A2DP while watching a video.
2. Audio / video synchronization and quality should be good.

18.5.12 AVRCP

18.5.12.1 AVRCP Media Control

1. Play/Pause, Next/Previous, FF/RW.
2. All controls should work when selected on the HMI.

18.5.12.2 Absolute Volume

1. Volume Up/Down.
2. Volume on Apple device and accessory should be synchronized.

18.5.12.3 AVRCP Settings Control

1. Shuffle, Repeat.
2. All controls should work when selected on the HMI.

18.5.12.4 Metadata

1. Switch between Music app, iTunes Radio and 3rd party apps.

2. Metadata should always work after each transition, no matter which app is being used.

18.5.12.5 AVRCP Browsing

1. Browse the folder structure and media from HMI. Select a track and a station in iTunes Radio to play.
2. Apple device's media folder structure should be browsable on HMI.

18.5.12.6 AVRCP Version

1. Connect accessory with phones that support higher AVRCP versions (1.5, 1.6, etc).
2. AVRCP functionalities should remain (Control, Metadata, browsing, etc).

18.5.13 Turn-by-turn Navigation

18.5.13.1 Turn-by-turn Navigation During A2DP Streaming

1. Stream A2DP. Start turn-by-turn navigation.
2. Turn-by-turn prompts should be audible during A2DP streaming. Prompts should be complete and sound clear.

18.5.13.2 Turn-by-turn Navigation when A2DP is Paused

1. Stream A2DP. Pause music. Start turn-by-turn navigation.
2. Turn-by-turn prompts should be audible with music in pause state. Prompts should be complete and sound clear.

18.5.13.3 Turn-by-turn Navigation Over HFP

1. Enable Maps Prompts over HFP. While listening to FM, start turn-by-turn navigation.
2. Virtual call should appear for each turn-by-turn prompt. Prompts should be complete and sound clear.

18.5.14 Phonebook

18.5.14.1 Contacts

1. Download phonebook from accessory.
2. Size and contents of the downloaded phonebook should match with Apple device phonebook.

18.5.14.2 Call History

1. Download call history from accessory.
2. Size and contents of the downloaded call history should match with Apple device call history (ich, och, mch).

18.5.14.3 Update

1. Add/Delete a contact/call history from Apple device. Download again from accessory.
2. Updated should be seen in the downloaded phonebook/call history on HMI.

18.5.15 iAP

The following reviews how to setup and capture iAP-over-Bluetooth with ATS for MFi accessory audits. This section covers only procedures and test cases for iAP1 and iAP2 over the Bluetooth transport. It does not cover any additional Bluetooth profile testing.

18.5.15.1 Equipment

To run iAP-over-Bluetooth accessory audits, you will need the following:

- Mac with ATS 4.1 or later
- ATS Utility 1.1.1 or later installed on an Apple device running iOS 8.0 or later
- ComProbe BPA 100
- RF shielding cloth
- Prop or stand for RF shielding cloth

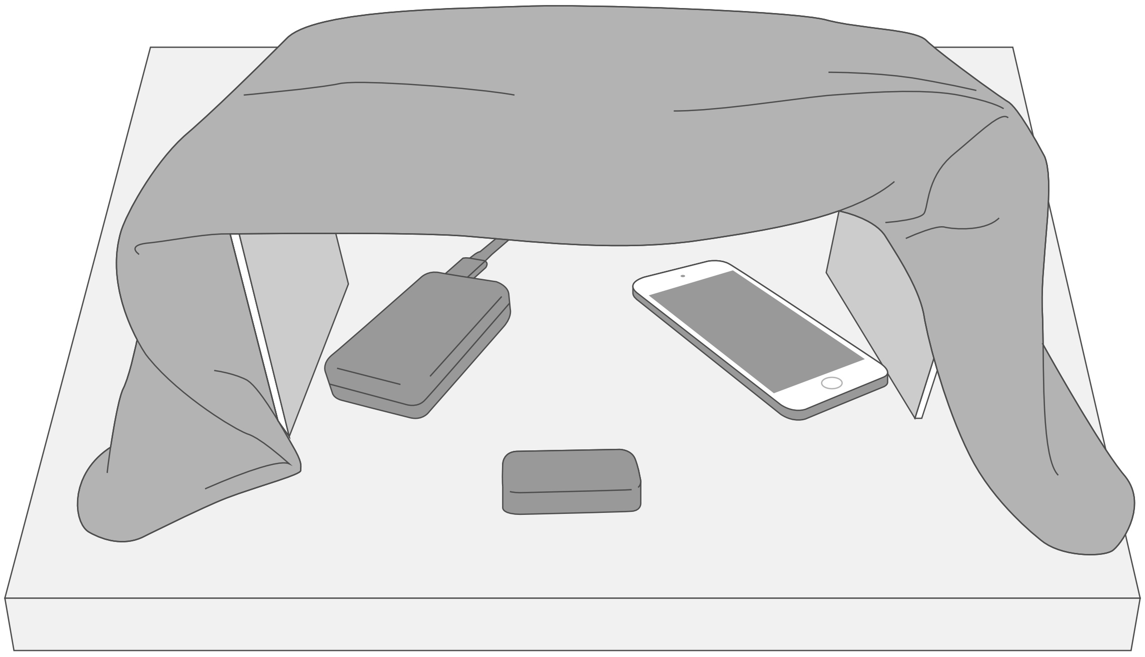
18.5.15.2 Setting up the test environment

Due to the inherent nature of capturing data wirelessly, ATS may not be able to always correctly capture the communication between the Apple device and accessory. If there is too much RF interference, or the Apple device and accessory are placed incorrectly with respect to the ComProbe you may see missing Bluetooth data in the ATS trace. To help alleviate these issues:

1. Make sure that the ComProbe, Apple device and the accessory are placed to form an equilateral triangle during capture.
2. Use RF cloth to shield the ComProbe, Apple device and the accessory from RF interference. Use a stand or prop to hold the cloth above the hardware. This will allow you to continue to interact with the Apple device and accessory during the capture.

3. Make sure that the RF environment is free of noise by turning off other devices that may use the 2.4 GHz frequency range such as Wi-Fi and other Bluetooth devices.

Figure 18-3 RF Interference Test Setup



18.5.15.3 Capturing iAP-over-Bluetooth

The first time you setup an iAP-over-Bluetooth capture, you will have to install drivers to your Mac. Additionally, you may have to update the firmware on your ComProbe.

1. Upon selecting the ComProbe in the Capture Configuration Assistant, you will be promoted to install the ComProbe driver. Select Install.
2. Run the installer to completion. Once the driver installation has completed you must restart your computer.
3. After restart, launch ATS and select the ComProbe. If the connected ComProbe contains older firmware, you will be prompted to update it. If promoted to do so, select Update.
4. Select "Install" to install the driver for the ComProbe firmware updater app.
5. The firmware updater app will launch in a separate window. Follow the on-screen instructions to update the ComProbe's firmware. This may take a few minutes.
6. When finished select "Done" and the firmware updater app will close. You are now ready to start capturing iAP-over-Bluetooth with ATS.

18.5.15.4 Capturing iAP-over-Bluetooth with ATS

Follow these instructions to setup an iAP-over-Bluetooth capture with the Capture Configuration Assistant. You may also setup Bluetooth captures from the Advanced Capture Configuration Assistant.

1. Select ComProbe BPA 100. Note: if more than one ComProbe is connected to your Mac, only one will be displayed in the Capture Configuration Assistant screen.
2. Select "New Bluetooth capture configuration".
3. Make sure your Apple device's Bluetooth is on and the device is discoverable. Select Start Inquiry to allow the ComProbe to look for all discoverable Bluetooth devices. Select your Apple device from the list of devices.
4. Make sure your accessory's Bluetooth is on and the device is discoverable. Select "Start Inquiry" to allow the ComProbe to look for all discoverable Bluetooth devices. Select your accessory from the list of devices.
5. Follow the on-screen instructions. Note that only accessories that use iAP will show a Link Key in ATS Utility app.
6. Select the iAP version the accessory supports.
7. Follow the on-screen instructions.
8. After selecting "Start Capture", switch the Apple devices Bluetooth to "On" and tap on the accessory's name to Connect. You will see iAP traffic in the capture window. Continue to interact with the Apple device and accessory as you would in accessory audits with wired accessories.

18.5.15.5 Special Considerations

Some accessories will only utilize iAP over Bluetooth for specific accessory interactions (for example, EA Session over iAP). For accessories such as these, a Link Key will not be viewable in ATS Utility until the iAP communication is initialized.

This behavior has been observed in automotive head units, speakers and receivers.

If you are unable to view a Link Key for an accessory which claims support for both iAP over Bluetooth and EA Session, please attempt the following.

1. After connecting the device to the accessory, launch the associated application and use the app.
2. Reopen ATS Utility and check for a Link Key.
3. With the Link Key, proceed with BT audit as instructed above.

Note that, similar to the way you will not get a Link Key until the EA Session has been triggered, you will not see iAP traffic until the application is launched.

18.5.15.6 EA string matching

Verify that the application protocol strings match the accessory protocol strings SDK Apps (page 540).

18.5.15.7 Autonomous app launch

Verify that the accessory only sends RequestAppLaunch (page 820)after the user has initiated the launch of the iOS app. A nag screen asking the user to "Allow" the use of the app must be presented after direct user action.

1. Verify that RequestAppLaunch (page 820) is only sent from the accessory in response to direct user action, such as pushing a button on the accessory App Launch (page 338). Do note that the automotive head units do not respond the same way.
2. If the accessory is iAP2-over-Bluetooth and sends RequestAppLaunch (page 820) LaunchAlert, verify that AppLaunchMethod is set to 0.

18.5.15.8 Power parameter test cases

For Bluetooth accessories that do not draw or provide power to an Apple device, verify that the accessory does not claim support for power via IdentificationInformation messages.

- Table 59-10 (page 810) must be set to None.
- MaximumCurrentDrawnFromDevice parameter must be set to 0. IdentificationInformation (page 807).

18.5.15.9 Standard test cases

Execute all additional applicable test procedures within this document.

19. Bluetooth Accessory Identification

This chapter describes Apple-specific Bluetooth commands that extend accessory capabilities beyond those supported by standard Bluetooth profiles.

To enable Apple-specific features, the accessory must support HFP Command AT+XAPL (page 370), which provides accurate information about the accessory's supported features. The Apple device will use the information sent by this command to enable and disable custom commands.

The accessory must send the following AT+XAPL command after making a successful HFP Service Level Connection (SLC) to the Apple device. The accessory must send an AT+XAPL command first, before sending any of the additional Apple-specific commands described below.

19.1 HFP Command AT+XAPL

Description: Enables custom AT commands from an accessory.

Initiator: Bluetooth accessory

Format: AT+XAPL=*vendorID*-*productID*-*version*,*features*

Parameters:

- *vendorID*: A string representation of the hex value of the vendor ID from the manufacturer, without the `0x` prefix.
- *productID*: A string representation of the hex value of the product ID from the manufacturer, without the `0x` prefix.
- *version*: The revision of the software.
- *features*: A base-10 representation of a bit field. Available features are:
  - Bit 0 = reserved
  - Bit 1 = The accessory supports battery reporting (reserved only for battery operated accessories).
  - Bit 2 = The accessory is docked or powered (reserved only for battery operated accessories).
  - Bit 3 = The accessory supports Siri status reporting.
  - Bit 4 = the accessory supports noise reduction (NR) status reporting.

- All other values are reserved.

Example: AT+XAPL=ABCD-1234-0100,10 (Supports battery reporting and Siri status)

Response: +XAPL=iPhone,*features*

20. Bluetooth Connection

Accessories can automate the Bluetooth pairing process between an Apple device and any number of integrated Bluetooth accessory components if they also have a wired connection to the Apple device.

Additionally, some Apple devices can report changes in the Bluetooth connection status for an accessory's Bluetooth components. These notifications can be useful when the accessory wishes to determine whether it is connected to a particular Apple device via both Bluetooth and a Lightning connector. Accessories must implement this feature if they can maintain audio transport connections over both Bluetooth A2DP and other transports (see Multiple Audio Connections (page 528)).

20.1 Bluetooth Connection Requirements

The accessory must declare at least one Bluetooth component during identification to use this feature. The component(s) must comply with the requirements in Bluetooth (page 344).

The accessory must send an initial BluetoothComponentInformation (page 822) message within 2 seconds of receiving a IdentificationAccepted (page 816) message. Additionally, the accessory must also send a BluetoothComponentInformation (page 822) message to the device whenever the state of an identified accessory Bluetooth component changes. The accessory must report status for every identified Bluetooth component and not just status for the changed component. Conversely, after the initial BluetoothComponentInformation (page 822) message is sent, the accessory must not send additional ones unless there is a state change in one of its identified Bluetooth components.

In the case of Bluetooth components, the accessory must report the `ComponentIdentifier` of the identified component and `ComponentEnabled` parameter value of `true` if the Bluetooth component is ready for connections. Conversely, the accessory must send a BluetoothComponentInformation (page 822) message with `ComponentEnabled` set to `false` if the Bluetooth component is not ready for connections.

All accessories that support the Bluetooth Connection feature via iAP2 must send or receive the following iAP2 control session message(s):

- BluetoothComponentInformation (page 822)
- StartBluetoothConnectionUpdates (page 822)
- BluetoothConnectionUpdate (page 823)
- StopBluetoothConnectionUpdates (page 824)

20.2 Bluetooth Connection Usage

Pairing of Bluetooth components with an Apple device will automatically initiate if the accessory has successfully authenticated and the Apple device has not previously paired with the component. The accessory's Bluetooth component must be prepared to complete the pairing process as soon as the accessory has started identification.

The accessory must send a StartBluetoothConnectionUpdates (page 822) message to start the generation of component updates from the device. More than one component identifier may be specified if the accessory has multiple Bluetooth components with different MAC addresses. If the accessory later wishes to stop receiving component status notifications it may send a StopBluetoothConnectionUpdates (page 824) message to the device.

When updates are first started, one or more initial BluetoothConnectionUpdate (page 823) messages will be sent to the accessory to inform it of the starting connection state of all requested components. Subsequent notifications for a particular component always completely override prior ones. It is possible for the device to connect to the same component profile multiple times in a row. For example, this may occur when the device temporarily moves out of range for a period of time that is not long enough to trigger an automatic device disconnect and reconnects upon rediscovering the accessory component. Therefore, accessories must handle duplicate BluetoothConnectionUpdate (page 823) messages gracefully.

Accessories must send StopBluetoothConnectionUpdates (page 824) when they no longer have a need for them. For example, if an accessory makes use of Bluetooth connection updates purely to automate pairing of the accessory's Bluetooth component, it must stop asking for updates and not start them again upon subsequent connections to the Apple device.

20.3 Test Procedures

20.3.1 iAP2 Tests

1. Verify that accessories which maintain audio transport connections over both Bluetooth A2DP and other transports implement this feature.
2. Verify that the following iAP2 control session message(s) are sent or received:
    - BluetoothComponentInformation (page 822)
    - StartBluetoothConnectionUpdates (page 822)
    - BluetoothConnectionUpdate (page 823)
    - StopBluetoothConnectionUpdates (page 824)

21. Bluetooth Headset Battery Level Indication

Any Hands-Free Bluetooth headset accessory can show its battery level to the user as an indicator icon in the Apple device status bar. This feature is supported on all Apple devices that support the Hands-Free Profile, including iPhone, iPod touch, and iPad.

Headset battery indication is implemented by two Apple-specific Bluetooth HFP AT commands, HFP Command AT+XAPL (page 370) and HFP Command AT+IPHONEACCEV (page 374)

21.1 HFP Command AT+IPHONEACCEV

Description: Reports a headset state change.

Initiator: Headset accessory

Format: `AT+IPHONEACCEV=`*Number of key/value pairs* `,` *key1* `,` *val1* `,` *key2* `,` *val2* `,...`

Parameters:

- *Number of key/value pairs* : The number of parameters coming next.
- *key* : the type of change being reported:
  - 1 = Battery Level
  - 2 = Dock State
- *val* : the value of the change:
  - Battery Level: string value between '`0`' and '`9`'
  - Dock State: `0` = undocked, `1` = docked

Example: `AT+IPHONEACCEV=1,1,3`

22. Bluetooth Low Energy

The *Bluetooth 4.0* specification introduces Bluetooth Low Energy, a new wireless technology targeted for accessories with limited battery resources. If Bluetooth Low Energy is supported, the accessory must follow the guidelines in this section.

22.1 Role

The accessory must implement either the Peripheral role as defined in the *Bluetooth 4.0* specification, Volume 3, Part C, Section 2.2.2.3 or the Broadcaster role, as defined in Section 2.2.2.1.

22.2 Advertising Channels

The accessory must advertise on all three advertising channels (37, 38, and 39) at each advertising event. See the *Bluetooth 4.0* specification, Volume 6, Part B, Section 4.4.2.1.

22.3 Advertising PDU

The accessory must use one of the following advertising PDUs:

- ADV_IND
- ADV_NOCONN_IND
- ADV_SCAN_IND

ADV_DIRECT_IND must not be used. See the *Bluetooth 4.0* specification, Volume 6, Part B, Section 2.3.1.

22.4 Advertising Data

The advertising data sent by the accessory must contain at least the following information as described in the *Bluetooth Core Specification Supplement*, Part A:

- Flags

- TX Power Level
- Local Name
- Services

The Local Name must match the accessory's markings and packaging and not contain ':' or ';'.

The accessory may put the Local Name and the TX Power Level data in the SCAN_RSP PDU if, for example, it needs to reduce power consumption or not all of the advertising data fit into the advertising PDU. Note that, depending on its state, the Apple product may not always perform active scanning.

The primary services must always be advertised in the advertising PDU. Secondary services must not be advertised. Services not significant to the primary use case of the accessory may be omitted if space is limited in the Advertising PDU.

The advertising data and the scan response data in the SCAN_RSP PDU must comply with the formatting guidelines in the *Bluetooth 4.0* specification, Volume 3, Part C, Section 18: it starts with a length field, followed by AD Type and AD Data.

22.5 Advertising Interval

The advertising interval of the accessory must be carefully considered, because it affects the time to discovery and connect performance. For a battery-powered accessory, its battery resources must also be considered.

To be discovered by the Apple product, the accessory must first use the recommended advertising interval of 20 ms for at least 30 seconds. If it is not discovered within the initial 30 seconds, Apple recommends using one of the following longer intervals to increase chances of discovery by the Apple product:

- 152.5 ms
- 211.25 ms
- 318.75 ms
- 417.5 ms
- 546.25 ms
- 760 ms
- 852.5 ms
- 1022.5 ms
- 1285 ms

Note: Longer advertising intervals usually result in longer discovery and connect times.

22.6 Connection Parameters

The accessory is responsible for the connection parameters used for the Low Energy connection. The accessory must request connection parameters appropriate for its use case by sending an L2CAP Connection Parameter Update Request at the appropriate time. See the *Bluetooth 4.0* specification, Volume 3, Part A, Section 4.20 for details. The connection parameter request may be rejected if it does not comply with all of these rules:

- *Interval Max * (Slave Latency + 1) ≤ 2 seconds*
- *Interval Min ≥ 20 ms*
- *Interval Min + 20 ms ≤ Interval Max*
- *Slave Latency ≤ 4*
- *connSupervisionTimeout ≤ 6 seconds*
- *Interval Max * (Slave Latency + 1) * 3 < connSupervisionTimeout*

If Bluetooth Low Energy HID is one of the connected services of an accessory, connection interval down to 11.25 ms may be accepted by the Apple product.

The Apple product will not read or use the parameters in the Peripheral Preferred Connection Parameters characteristic. See the *Bluetooth 4.0* specification, Volume 3, Part C, Section 12.5.

22.7 Privacy

The accessory must be able to resolve a Resolvable Private Address in all situations. Due to privacy concerns, the Apple product will use a Random Device Address as defined in the *Bluetooth 4.0* specification, Volume 3, Part C, Section 10.8.

22.8 Permissions

The accessory must not require special permissions, such as pairing, authentication, or encryption to discover services and characteristics. It may require special permissions only for access to a characteristic value or a descriptor value. See the *Bluetooth 4.0* specification, Volume 3, Part G, Section 8.1, fifth paragraph.

22.9 Pairing

The accessory must not request pairing until an ATT request is rejected using the Insufficient Authentication error code. See the *Bluetooth 4.0* specification, Volume 3, Part F, Section 4 for details.

If, for security reasons, the accessory requires a bonded relationship with the Central, the Peripheral must reject the ATT request using the Insufficient Authentication error code, as appropriate. As a result, the Apple product may proceed with the necessary security procedures.

Similarly, if the Apple device acts as a Central and a GATT server, it may reject an ATT request using the Insufficient Authentication error code. The accessory must initiate the security procedure for pairing in response.

Pairing may require user authorization depending on Apple product. Once an accessory is paired with an Apple product, it must retain the distributed keys of both central and peripheral for future use. If the pairing is no longer required, the accessory must delete both sets of keys.

22.10 MTU Size

The Apple device supports and requests MTU size larger than default MTUs during the Exchange MTU Request handshake. See the *Bluetooth 4.0* specification, Volume 3, Part F, Section 3.2.8. The accessory may use this mechanism to negotiate larger MTU sizes.

22.11 Services

22.11.1 Generic Access Profile Service

The accessory must implement the Device Name characteristic per the *Bluetooth 4.0* specification, Volume 3, Part C, Section 12.1. The Device Name characteristic must be writeable.

22.11.2 Generic Attribute Profile Service

The accessory must implement the Service Changed characteristic only if the accessory has the ability to change its services during its lifetime.

The Apple product may use the Service Changed characteristic to determine if it can rely on previously read (cached) information from the device. See the *Bluetooth 4.0* specification, Volume 3, Part G, Section 7.1.

22.11.3 Device Information Service

The accessory must implement the Device Information Service. The service UUID for this service must not be advertised in the Advertising Data. The following characteristics must be supported:

- Manufacturer Name String
- Model Number String
- Firmware Revision String
- Software Revision String

22.11.4 Available Services

With iOS 7.0, any Apple device makes Battery Service, Current Time Service and Apple Notification Center Service (ANCS) available to an accessory. The Current Time Service supports the current time and local time information characteristics. The service does not provide an "Adjust Reason" when the current time changes. ANCS uses 7905F431-B5CE-4E99-A40F-4B1E122D00D0 as its UUID.

These services are not guaranteed to be available immediately after connection and the accessory must support Characteristic Value Indication of the Service Changed characteristic (see *Bluetooth 4.0* specification, Volume 3, Part G, Section 7.1) to be notified when the services become available. The Apple device will maintain a connection to an accessory as long as it is paired and uses one of the available services.

22.12 GATT Server

With iOS 6.0, applications may contribute services and characteristics to the GATT server that the Apple device makes available to the accessory. The recommendations in this section apply to the accessory in this case.

The following services are implemented internally by iOS and must not be published by third party iOS applications:

- Generic Attribute Profile Service
- Generic Access Profile Service
- Bluetooth Low Energy HID Service
- Battery Service
- Current Time Service
- Apple Notification Center Service

The Apple device implements the GAP Service Changed characteristic, because the database contents can change at any time. The accessory must therefore support the Characteristic Value Indication of this characteristic and, upon receiving indications, invalidate its database cache accordingly. See the *Bluetooth 4.0* specification, Volume 3, Part G, Section 7.1.

The accessory must minimize the use of ATT/GATT requests and commands and only send what is necessary. For example, do not use GATT Discover All Services when the accessory is looking for specific services. Use Discover Primary Service By Service UUID instead. Less airtime equals less power consumption and better performance for both the accessory and the Apple device.

When third party iOS applications discover services on the accessory, the following services are used internally by iOS and are filtered out from the list of discovered services:

- Generic Attribute Profile Service
- Generic Access Profile Service
- Bluetooth Low Energy HID Service
- Apple Notification Center Service

The accessory must be robust enough to handle any error gracefully. Pairing and Characteristic Value reads/writes may fail if the application that owns the service is not in the foreground and is not entitled to run in the background.

If an ATT Prepare Write Request is used, all queued attributes are contained within the same GATT Service.