Bluetooth Accessory Design Guidelines for Apple Products

Release R7

Developer

Contents

Figures and Tables

1. Introduction

This document presents design guidelines for hardware accessories that use Bluetooth transport to communicate with Apple products, including Mac, iPhone, iPad, and iPod touch models.

To be compatible with Apple products, both current and future, Bluetooth accessories should follow the specifications in this document. An Apple product may make feature availability contingent on the Bluetooth accessory following the specifications in this document.

> **Note:** This document uses the term "Apple product" to refer generically both to Mac (Apple computers that run OS X) and to iPod, iPhone, and iPad models. Among the latter products, those that run iOS (Apple's mobile operating system) are also referred to as "iOS devices." Specifications in this document that are designated for iOS devices apply only to those products.

1.1 Organization of This Document

The specifications in this document are presented as follows:

- "2. Bluetooth" (page 7) relates the design of hardware accessories to the general Bluetooth specification.
- "3. Bluetooth Low Energy" (page 20) relates the design of hardware accessories to the general Bluetooth Low Energy specification.
- "4. Bluetooth Accessory Identification" (page 26) describes enabling Apple-specific Bluetooth commands.
- "5. Bluetooth Headset Battery Level Indication" (page 28) describes providing headset battery level information to an Apple device.
- "6. Siri" (page 29) presents design guidelines for Bluetooth accessories that will use the Siri functionality of iOS devices.
- "7. iPod Accessory Protocol" (page 37) references an Apple protocol that can extend accessory capabilities beyond those supported by standard Bluetooth profiles.

1.2 Apple Bluetooth Development Mailing List

Questions or comments regarding accessory development for Apple products can be posted to the Apple mailing list for Bluetooth development, bluetooth-dev@lists.apple.com. To join the mailing list, visit lists.apple.com/mailman/listinfo/bluetooth-dev.

Engineers on the Apple Bluetooth development team monitor this mailing list and will try to answer your questions. Please search the archives first, to see if your topic has already been discussed.

2. Bluetooth

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

2.1 Conformity With Bluetooth Specifications

Every accessory that is compatible with an Apple product shall 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.

2.1.1 Enhanced Data Rate

The Enhanced Data Rate (EDR) feature introduced in the Bluetooth 2.0 specification enables accessories to communication more efficiently. Every accessory shall 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 WiFi and other connected Bluetooth devices because it frees up more air time.
- It improves performance in multipoint configurations.

2.1.2 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:

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

Accessories that are compatible with Apple products should 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 WiFi.

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 shall accept the request and its parameters without negotiation.

If the accessory sets the sniff mode parameters, the accessory shall 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 should 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.

2.1.3 Role and Topology Management

Every accessory that is compatible with an Apple product shall:

- 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 shall 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 WiFi, 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 should 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 should ask to be master at that time.

2.1.4 Extended Inquiry Response

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

- The Local Name of the product (Complete or Shortened).
- The TX Power Level.
- The Service Class UUID for the iAP protocol, if the product has this service. For information about iAP, see "7. iPod Accessory Protocol" (page 37).

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.

2.1.5 Secure Simple Pairing

Every accessory that is compatible with an Apple product shall:

- 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 should be used whenever feasible. See Volume 1, Section 5.4 in the *Bluetooth Core Specification*, Version 2.1 + EDR.

2.2 Profiles

The Apple knowledge base article support.apple.com/kb/ht3647 provides a complete list of the Bluetooth profiles that certain iOS 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.

2.2.1 Device ID Profile (DID)

Every accessory that is compatible with an Apple product shall:

- Support the Bluetooth Device ID Profile, version 1.3 or higher.

- Use the company identifier from the Assigned Numbers document assigned by the Bluetooth SIG as its Vendor ID value (VID). See 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 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.

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 should 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 should have different Product IDs. The accessory can also use a secondary Device ID to uniquely identify the product ID or model number.

2.2.2 Hands-Free Profile (HFP)

If a accessory supports the Bluetooth *Hands-Free Profile* specification, it shall be version 1.5 or higher.

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

2.2.2.1 Remote Audio Volume Control

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

- 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.

2.2.2.2 Indicator Event Reporting

Every accessory that is compatible with an Apple product and supports HFP should 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 should 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 should request the initial status using the `AT+CIND=?` and `AT+CIND?` commands, according to the HFP specification.

2.2.2.3 Voice Recognition Activation

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

- 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.

2.2.2.4 Echo Cancellation and Noise Reduction

When echo cancellation and noise reduction are performed locally on a Hands-Free accessory, it should 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.

2.2.2.5 In-Band Ringing

Every accessory that is compatible with an Apple product and supports HFP should 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 should sound on the hands-free accessory.

2.2.2.6 Synchronous Connection

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

- 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 should 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 WiFi 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.

2.2.2.7 Wide Band Speech

Every accessory that is compatible with an Apple product and supports HFP should 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 should support the T2 link parameter settings.

All iOS devices running iOS 5 or later support Wide Band Speech. If both the iOS 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.

2.2.3 Message Access Profile (MAP)

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

- 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. iOS devices do not support sending messages using MAP.

All iOS devices running iOS 6.0 or later support MAP.

2.2.4 Audio/Video Remote Control Profile (AVRCP)

To support the *Audio/Video Remote Control Profile* , a accessory that is compatible with an Apple product should support the buttons and operations listed in this section.

2.2.4.1 Supported Buttons

Every accessory that is compatible with an Apple product and supports the *Audio/Video Remote Control Profile* should use separate button commands to play and pause instead of toggling the play or pause state.

2.2.4.2 Supported Operations

Apple products support the following operation_IDs in Pass Through commands:

- Play
- Stop
- Pause
- Fast Forward
- Rewind
- Forward
- Backward

2.2.4.3 Repeat and Shuffle Modes

Every iOS 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 iOS 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.

2.2.4.4 Notifications

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

Every iOS 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

2.2.4.5 Volume Handling

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

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

2.2.4.6 Browsing

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

- Not try to index or cache the entire library upon connection. The iOS device 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 iOS devices running iOS 6.0 or later support AVRCP Browsing.

2.2.4.7 IOS App-Provided Metadata

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

2.2.5 Advanced Audio Distribution Profile (A2DP)

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

2.2.5.1 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 2-1 (page 15).

Table 2-1 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 should support 53.

2.2.5.2 MPEG 2/4 AAC Codecs

iOS devices support the non-mandatory codec MPEG-2/4 AAC, as defined in Section 4.5 of the A2DP specification, Version 1.2. Accessories should 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 iOS devices are listed in Table 2-2 (page 16).

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

Element	Value
Object Type	MPEG-2 AAC LC
Sampling Frequency	44,100 Hz
Channels	2
Bit rate	264,630 bps
VBR	0

AAC audio stream packets in iOS devices have the structure shown in Table 2-3 (page 16).

Table 2-3 AAC audio packet for iOS 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 2-3 (page 16).

- The suggested L2CAP MTU value for each iOS 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 *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 should support VBR capability. The iOS device will be varying AAC bit rate depending on the content and the accessory should be able to handle the variation without causing gap in the audio.

2.3 Audio Routing

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

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

- HFP using eSCO channel
- A2DP using ACL channel

The iOS 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 iOS device.

2.3.1 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 should not handle any other audio content.

2.3.2 Audio Data Received via A2DP Profile

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

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

2.3.2.1 Differentiating Audio Content from System Sounds

Music-like content can be differentiated from system sound by adding support for the AVRCP profile version 1.3 or later. The AVRCP profile allows an accessory to be aware of the audio playback state in the iOS device, using notifications.

When an iOS 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 *AVRCP* specification, version 1.4. This indicates that audio data via the A2DP profile contains music. When an iOS device initiates audio playback over an A2DP channel for playing system sound, no AVRCP notification is sent.

Figure 2-1 (page 18) and Figure 2-2 (page 19) show the difference between the notifications for music playback and for system sounds.

Figure 2-1 Initiate Audio Playback (e.g. music)

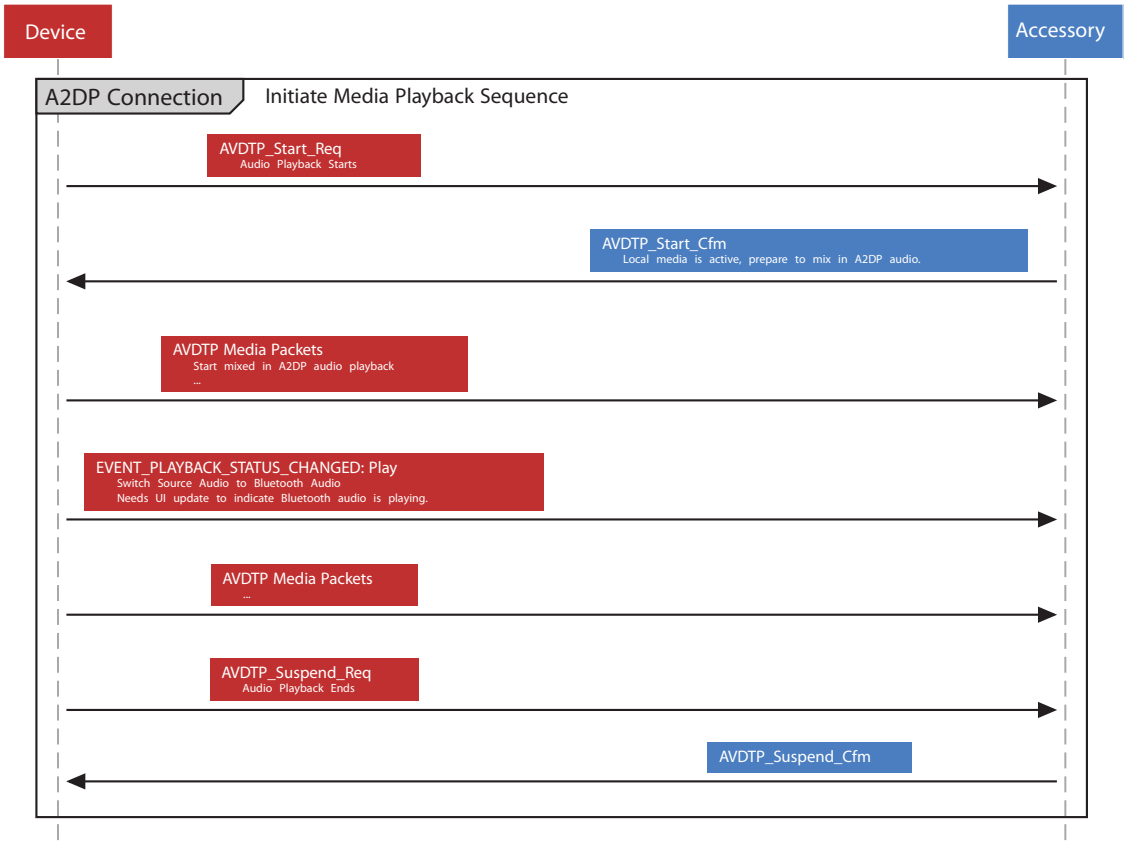
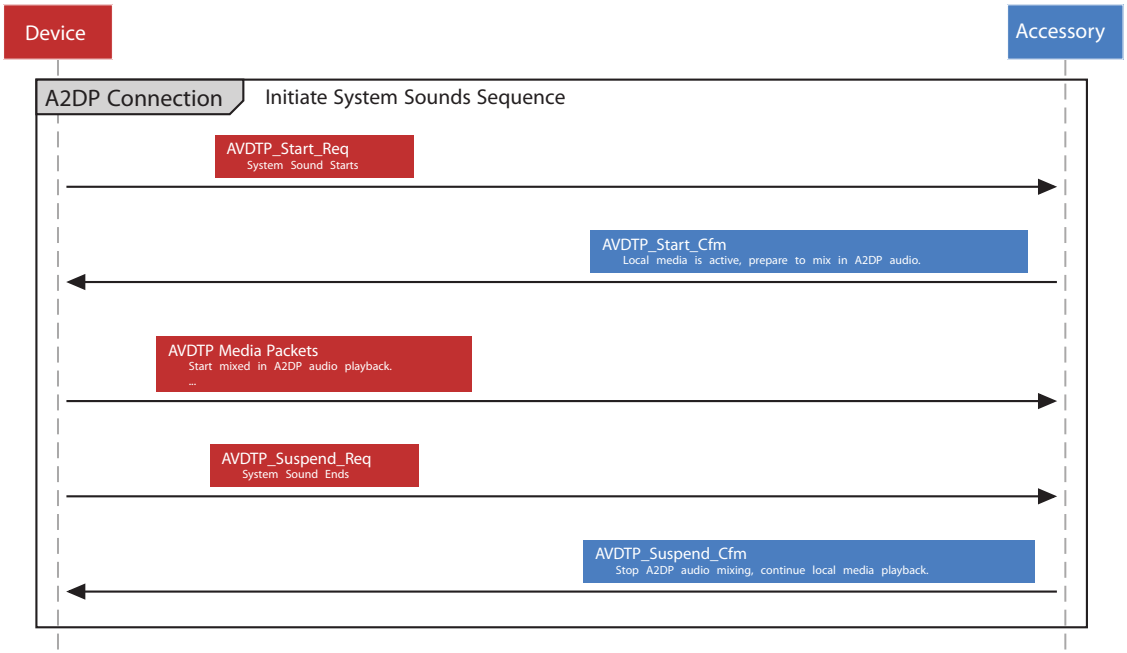


Figure 2-2 Initiate System Sound (e.g. turn-by-turn directions)



2.3.2.2 Expected Audio Routing Behavior for A2DP

The accessory should 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.

3. 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 should follow the guidelines in this section.

3.1 Role

The accessory should 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.

3.2 Advertising Channels

The accessory should 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.

3.3 Advertising PDU

The accessory should use one of the following advertising PDUs:

- ADV_IND
- ADV_NOCONN_IND
- ADV_SCAN_IND

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

3.4 Advertising Data

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

- Flags

- TX Power Level
- Local Name
- Services

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 should always be advertised in the advertising PDU. Secondary services should 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 should 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.

3.5 Advertising Interval

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

To be discovered by the Apple product, the accessory should 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.

3.6 Connection Parameters

The accessory is responsible for the connection parameters used for the Low Energy connection. The accessory should 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.

3.7 Privacy

The accessory should 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.

3.8 Permissions

The accessory should 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.

3.9 Pairing

The accessory should 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 should 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 iOS device acts as a Central and a GATT server, it may reject an ATT request using the Insufficient Authentication error code. The accessory should 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 shall retain the distributed keys of both central and peripheral for future use. If the pairing is no longer required, the accessory shall delete both sets of keys.

To incorporate authentication in an accessory design, the accessory should support the Authentication Specification of MFi program. For further information about MFi, see developer.apple.com/programs/mfi.

3.10 MTU Size

The iOS 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.

3.11 Services

3.11.1 Generic Access Profile Service

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

3.11.2 Generic Attribute Profile Service

The accessory shall 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.

3.11.3 Device Information Service

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

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

3.11.4 Available Services

With iOS 7.0, any iOS product makes Battery Service, Time Service and Apple Notification Center Service (ANCS) available to an accessory. The 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 shall 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 iOS device will maintain a connection to an accessory as long as it is paired and uses one of the available services.

3.12 GATT Server

With iOS 6.0, applications may contribute services and characteristics to the GATT server that the iOS 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 shall not be published by third party iOS applications:

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

The iOS device implements the GAP Service Changed characteristic, because the database contents can change at any time. The accessory should 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 should 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 should 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.

4. 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 "4.1 HFP Command AT+XAPL" (page 26), 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 should send an AT+XAPL command first, before sending any of the additional Apple-specific commands described below.

4.1 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*

5. 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 iOS device status bar. This feature is supported on all iOS 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, "4.1 HFP Command AT+XAPL" (page 26) and "5.1 HFP Command AT+IPHONEACCEV" (page 28)

5.1 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`

6. Siri

6.1 Enabling Custom Siri Commands

Every accessory that works with Siri must support "4.1 HFP Command AT+XAPL" (page 26). The iOS device will use the information sent by this command to enable and disable custom commands related to Siri.

To receive Siri status events, the accessory must send the `AT+XAPL` command after making a successful HFP Service Level Connection (SLC) to the iOS device. The accessory should send an `AT+XAPL` command first, before sending any of the additional Siri-specific commands described below.

6.2 Obtaining Siri Availability Information

After establishing an HFP profile connection, an accessory can determine if Siri is available and enabled on an iOS device. It can also receive notifications of changes in Siri status. If Siri is disabled, Voice Control will be activated instead.

6.2.1 Obtaining Status Information at Connection

The accessory should send the following command after making a successful HFP profile (SLC) connection and sending an `AT+XAPL` command.

6.2.1.1 HFP Command AT+APLSIRI?

Description: AT command to retrieve Siri status information.

Initiator: accessory

Format: `AT+APLSIRI?`

Response: `+APLSIRI:`*value*

Defined Values:

- 0 = Siri is not available on this platform.
- 1 = Siri is available and enabled.
- 2 = Siri is available but not enabled.

Example: `+APLSIRI:1` (Siri is available and enabled)

6.2.2 Receiving Siri Availability Updates from the iOS Device

After initialization has been completed, the iOS device will send the accessory the following notification if there is a change in Siri status. This notification will be provided only if the accessory has requested Siri status (by sending `AT+APLSIRI?`) at least once after connection, and if the iOS device has reported that Siri is available and enabled.

6.2.2.1 HFP Command +APLSIRI

Description: Unsolicited event indicating a change in Siri status.

Initiator: iOS device

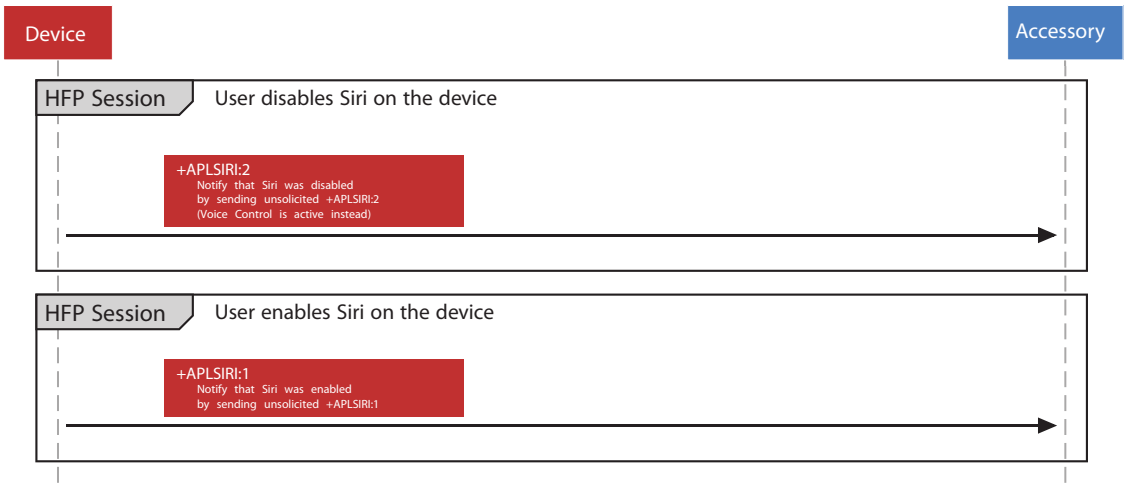
Format: `+APLSIRI:`*value*

Defined Values:

- 1 = Siri is available and enabled.
- 2 = Siri is available but not enabled.

Example: `+APLSIRI:2` (Siri is available but not enabled)

Figure 6-1 Siri is Disabled/Enabled from the iOS Device's Settings



6.3 Initiating a Siri Session

Once support for Siri is established on both the accessory and the iOS device, a Siri session can be started from either one.

6.3.1 Initiating a Session from the Accessory

To initiate a Siri session, the accessory must use the voice recognition command `AT+BVRA` defined in the Bluetooth *Hands-Free Profile* specification. For further details, see the Bluetooth *Hands-Free Profile* 1.6 profile specification, section 4.25. The HFP profile must be connected and SLC must exist.

The accessory should use the following command sequence:

- The accessory sends an `AT+BVRA=1` command to the iOS device.
- The iOS device sends an `OK` response.
- The iOS device launches a Siri session and creates a Synchronous Connection (SCO) for the audio.
- If the Siri session is not finished, the accessory must send `AT+BVRA=1` to continue the conversation. This may need to happen multiple times.
- When the Siri session is finished, the iOS device sends a `+BVRA:0` result code to the accessory.
- The iOS device disconnects the SCO connection.

While a Siri session is active, the accessory must let the user continue the conversation and ask follow up questions within the current context. In order to do so, the accessory must be able to send an AT+BVRA=1 command to the iOS device even after Siri has been already activated and before +BVRA:0 is received. Figure 6-2 (page 32) shows an overview of the interaction when Siri is triggered from the accessory, the running session was continued twice and once Siri was finished, the device dismissed the session.

Figure 6-2 Initiating a Siri Session from the Accessory

6.3.2 Initiating a Session from the iOS Device

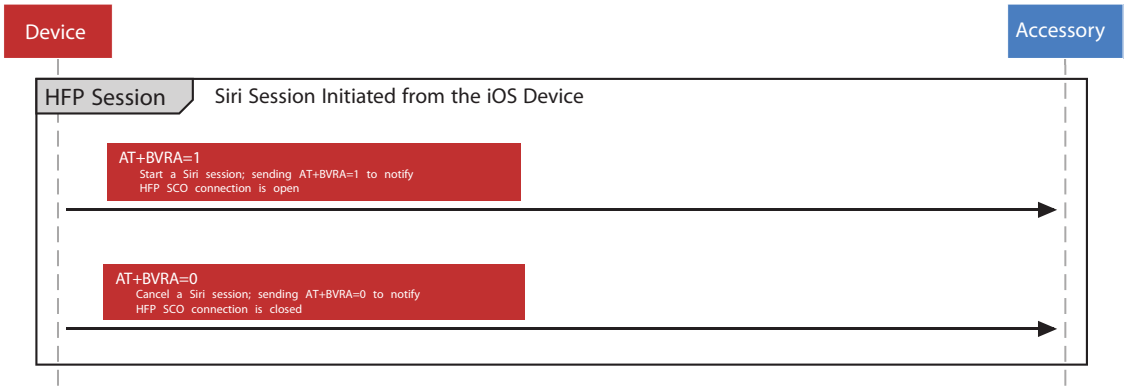
If the accessory supports voice recognition commands, the iOS device sends a +BVRA event to indicate the start of a Siri session. The accessory must enable support for voice recognition and indicate it in its feature response as described in the Bluetooth *Hands-Free Profile* 1.6 specification, section 4.34.1, "Bluetooth Defined AT Capabilities." Specifically, the HFP profile must be connected, SLC must exist, and voice recognition activation (bit 3) must be enabled in the AT+BRSF command. The iOS device will not use virtual call functionality for the Siri session if voice recognition activation is supported by the accessory.

The accessory should expect the following command sequence:

- The iOS device sends a +BVRA:1 event to the accessory.

- The iOS device launches a Siri session and creates a SCO connection for the audio.
- When the Siri session is finished, the iOS device sends a `+BVRA:0` result code to the accessory.
- The iOS device disconnects the SCO connection.

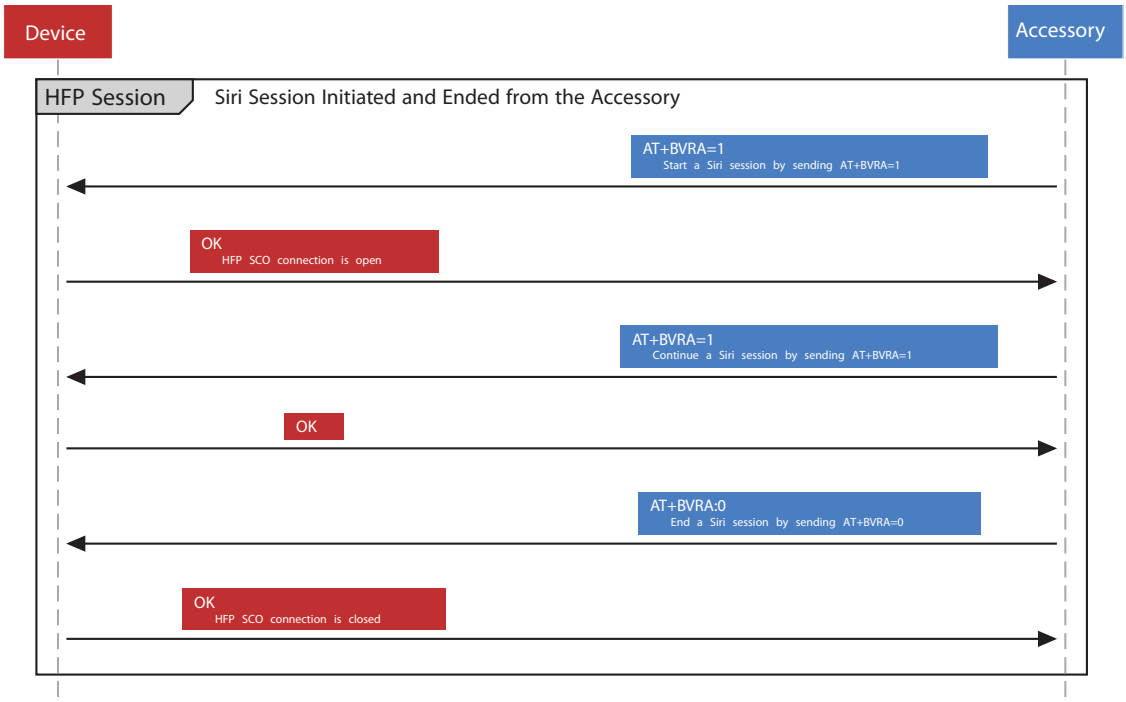
Figure 6-3 Initiating a Siri Session from the iOS Device



6.3.3 Ending a Session from the Accessory

Once a Siri session is running the accessory must be capable of ending the session by sending an `AT+BVRA=0` command to the iOS device. Figure 6-4 (page 34) shows an example of ending a running Siri session from the accessory. The accessory should only end an active session as a direct result of a user action.

Figure 6-4 Ending a Siri Session from the Accessory



6.4 Siri Eyes Free Mode

Siri Eyes Free mode is a feature to control Siri responses that include display information and can be enabled or disabled as needed. Siri Eyes Free mode is supported only for Bluetooth-enabled vehicle entertainment systems and should not be used by any other accessories.

The iOS device will listen for the HFP AT command `AT+APLEFM` to enable or disable Eyes Free mode.

This command is used by the iOS device to modify Siri responses that contain visual information or require user interaction. Suitable audio feedback and voice commands will be available to the user based on the Siri use-case that was initiated.

Eyes Free mode is disabled by default. Once the accessory has enabled Eyes Free mode, it remains enabled for all subsequent Siri sessions initiated from the accessory until the accessory disables it or the Bluetooth connection is disconnected.

6.4.1 HFP Command AT+APLEFM

Description: An accessory sends this command to notify an iOS device of the preferred state of Eyes Free mode.

Initiator: accessory

Format: `AT+APLEFM=`*value*

Response: `OK`

Defined Values:

- 0x00 = Disable Eyes Free mode.
- 0x01 = Enable Eyes Free mode.
- 0x02-0xFF = reserved

Example: `AT+APLEFM=1`

6.5 Improving Voice Recognition

The microphone audio that the accessory sends to the iOS device during a Siri session should be optimized for voice recognition rather than for human perception (such as during a cellular phone call).

Filtering of the audio signal to remove echoes or feedback noise is acceptable.

To provide the best possible audio quality as Siri input, the accessory must observe the following recommendations:

- **Echo cancellation and noise suppression (EC/NR)**: The accessory must perform echo cancellation and linear noise processing tuned for speech recognition, without excessive noise suppression that attenuates fricative onsets. The accessory must not use non-linear noise suppression algorithms because these can spectrally attenuate or shape frication.
- **Signal gain**: When adjusting signal levels, the accessory must avoid artifacts, dropouts, and clipping in all circumstances. Automatic Gain Control is not recommended. If the accessory adjusts signal gain, the gain should be held constant across each spoken utterance. A recommended target is to maintain audio signal levels such that the peaks are -15 dBFS.
- **Signal-to-noise ratio (SNR)**: An average SNR greater than 20 dB is recommended. Below 20 dB, recognition rates will be impacted.
- **Reverberation**: Maintaining RT60 time at less than 200 msec is recommended.

6.5.1 Wide-Band Speech Support

An accessory using Siri should support 16 kHz wide-band speech audio for better audio quality and voice recognition performance. See the Bluetooth *Hands-Free Profile* 1.6 specification for details about wide-band speech audio. Narrow-band audio signal (8 kHz) is supported but not recommended.

6.6 Optimizing the Siri Session

For best results in using Siri, the accessory design should follow these guidelines:

- The start of a Siri session should not be accompanied by local beeps or verbal indications (such as an announcement of "...voice dialing...") from the accessory. When a Siri session becomes active, the iOS device sends two beeps indicating that Siri is ready to receive instructions. Adding extra audible notifications only inserts delays in the system.
- The accessory should wait for the iOS device to end each Siri session. The accessory should not send an `AT+BVRA=0` command unless it is prompted to do so by user interaction.
- The iOS device expects that the accessory is capable of rendering audio as soon as the SCO connection is active. This is necessary to make sure that the user always hears the Siri introductory beeps with minimum delay. The delay should be within 200 ms.

7. iPod Accessory Protocol

Third-party accessories can use the iPod Accessory Protocol (iAP) to access advanced features of iOS devices. One such feature is the ability to communicate securely with third-party iOS applications via the iOS External Accessory Framework. For information about the External Accessory Framework, see http://developer.apple.com/library/ios/#featuredarticles/ExternalAccessoryPT/Introduction/Introduction.html on the Apple iOS Developer site.

To incorporate iAP into an accessory design, the accessory developer must be a member of the Apple MFi licensing program and integrate specific MFi hardware into the accessory. For further information about MFi, see developer.apple.com/programs/mfi.

Document Revision History

This table describes the changes to *Bluetooth Accessory Design Guidelines for Apple Products*.

Date	Notes
2013-09-18	*Revision R7:* Reorganized document for new content. Clarified and added content to "2.3.2.1 Differentiating Audio Content from System Sounds." Created separate chapter for "3. Bluetooth Low Energy" and added content to it. Added Chapter "Siri."
2012-11-06	*Revision R6:* Added section "Audio Routing" Added section "Message Access Profile (MAP)" Added section "GATT Server" Added section "Browsing" Removed 'USB' from vendor and product ID descriptions. Made other corrections and updates.
2011-10-14	*Revision R5:* Moved "Headset Battery Level Indication" section to Apple Protocols chapter. Moved Low Energy chapter to Chapter 2. Added section "Wide Band Speech". Added section "Apple Bluetooth Development Mailing List".

Date	Notes
	Added section "Headset Battery Level Indication". Added new material to section "Audio/Video Remote Control Profile (AVRCP)". Clarified Remote Audio Volume Control and added Low Energy section. Made other corrections and updates.
2011-04-04	*Revision R4:* Changed specification language from requirements to guidelines.
2011-03-04	*Revision R3:* Added Mac as Apple products covered by this specification. Changed title of document from "Bluetooth Accessory Design Guidelines for iOS Devices" to "Bluetooth Accessory Design Guidelines for Apple Products." Revised definition of Vendor ID source in "Device ID Profile (DID)".
2010-11-30	*Revision R2:* Retitled document from *Designing Bluetooth Acccessories for iOS Devices* . Corrected generic references to Apple devices, iOS devices, and iPod throughout the document. Expanded specifications in section "Advanced Audio Distribution Profile (A2DP)". Made minor other corrections and updates.
2010-09-08	*Revision R1:* First release.

Apple Inc.
1 Infinite Loop
Cupertino, CA 95014
408-996-1010