23. CarPlay

This chapter defines how a vehicle infotainment system (head unit) may receive and interact with digital content (audio, video, and metadata) streamed from an Apple device running iOS. It covers accessories that can receive streamed digital content from Apple devices.

To work with Apple CarPlay™, accessories must support the CarPlay feature.

CarPlay is only allowed for dashboard-integrated vehicle infotainment systems. It is not intended for rear-seat or non-vehicle integrations. The accessory must be connected to the vehicle audio system and must use an external microphone.

CarPlay accessories must only be marketed in countries where the CarPlay feature is available. For a list of countries where the CarPlay feature is available, see http://www.apple.com/ios/feature-availability/#applecarplay-applecarplay.

All CarPlay accessories must comply with the *Apple CarPlay Identity Guidelines* .

In the context of CarPlay, the term accessory is used to refer to either a vehicle and its head unit or an after-market head unit.

After-market head units must be integrated systems with a built-in display with a minimum 6 inch diagonal screen size. After-market head units that rely on the customer, an installer, or another third party to ensure spec compliance should make those requirements clear in their installation manual, documentation, etc. Examples of this include instructions on connecting vehicle speed sensors and external GNSS antennas, microphone placement, and placement of the Apple device for thermal considerations.

23.1 Additional Specifications

For clarifications and detailed specifications, this chapter cites portions of the following external specifications:

- USB Specifications
  - USB 2.0 Specification (http://www.usb.org/developers/docs/usb20/)
  - Universal Serial Bus Communications Class Subclass Specifications for Network Control Model Devices (http://www.usb.org/developers/docs/devclass_docs/)
  - Universal Serial Bus Class Definitions for Communication Devices (http://www.usb.org/developers/docs/devclass_docs/)

- IETF RFCs
  - 793 (http://www.ietf.org/rfc/rfc793.txt)
  - 768 (http://www.ietf.org/rfc/rfc768.txt)
  - 2460 (http://www.ietf.org/rfc/rfc2460.txt)
  - 2474 (http://www.ietf.org/rfc/rfc2474.txt)
  - 3966 (http://www.ietf.org/rfc/rfc3966.txt)
  - 5905 (http://www.ietf.org/rfc/rfc5905.txt)
  - "Internet Protocol (IP), DARPA Internet Program, Protocol Specification", RFC 791, September 1981
  - "An Ethernet Address Resolution Protocol (ARP)", RFC 826, November 1982
  - "Unique Local IPv6 Unicast Address", RFC 4193, October 2005
  - "Dynamic Host Configuration Protocol (DHCP)", RFC 2131, March 1997
  - "Internet Control Message Protocol (ICMP), DARPA Internet Program, Protocol Specification", RFC 792, September 1981
  - "Internet Control Message Protocol (ICMPv6) for the Internet Protocol Version 6 (IPv6) Specification", RFC 4443, March 2006
  - "Neighbor Discovery for IP version 6 (IPv6)", RFC 4861, September 1007
  - K. McCloghrie, "Management Information Base for Network Management of TCP/IP-based internets: MIB-II", RFC 1213, March 1991
  - F. Kastenholz, "Definition of Managed Objects for the Ether-like Interface Types", RFC 1398, January 1993
- IEEE 802.11 Standards
  - Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specification, IEEE Std. 802.11-2012
  - Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specification: Enhancements for Very High Throughput in Bands below 6 GHz, IEEE 802.11ac-2013
- Wi-Fi Alliance Specifications
  - Wi-Fi 802.11 with WPA2 System Interoperability Test Plan for IEEE 802.11a, b & g Devices, Version 2.4.2, March 2005
  - Wi-Fi 802.11n System Interoperability Test Plan, Version 2.0.1
  - Wi-Fi CERTIFIED(tm) ac Interoperability Test Plan, Version 1.0.0
  - Wi-Fi WMM System Interoperability Test Plan, Version 1.3.2, June 2006
  - Wi-Fi Voice Personal Interoperability Test Plan Version 2.3.3
- Bluetooth SIG Specifications

    - Specification of the Bluetooth System, "Bluetooth v2.1 + EDR", February 2007
    - "Bluetooth Specification Version 4.1", December 2013
    - Supplement to the Bluetooth Core Specification, Version 4, Dec 2013
- Apple Bonjour Support Site (https://www.apple.com/support/bonjour/)
- ITU Specifications
    - ITU H.264 Specification (http://www.itu.int/rec/T-REC-H.264)
    - ITU-T P.1100: Narrow-band hands-free communication in motor vehicles (http://www.itu.int/rec/T-REC-P.1100-201103-I/)
    - ITU-T P.1110: Wideband hands-free communication in motor vehicles (http://www.itu.int/rec/T-REC-P.1110-200912-I/)

23.2 General Requirements

23.2.1 Overview

Apple CarPlay lets an Apple device stream interactive digital content to an accessory with one or more audio/video endpoints and controls.

CarPlay is intended for use with a single front seat display. An accessory must not replicate digital video content to multiple displays.

For more information on how to integrate the Apple CarPlay identity in vehicle hardware and software and in marketing communications, refer to the *Apple CarPlay Identity Guidelines* .

23.2.2 Hardware Requirements

This section contains physical requirements for an accessory that supports CarPlay.

23.2.2.1 High Resolution Display

The accessory must provide a high-resolution display capable of rendering the User Interface (UI) stream without scaling and with all pixels visible to the end user. The display must have a minimum resolution of 800 x 480, support 24 bits of RGB color per pixel, and 30 Hz refresh rate (60 Hz recommended). Note that future specifications may require at least 60 Hz refresh rate. Displays with square pixels are recommended, but the Apple device can accommodate for non-square pixel displays as well. See User Interface (UI) Stream (page 411).

23.2.2.2 Processing

The accessory must be capable of:

- Hardware decoding of H.264 video, see User Interface (UI) Stream (page 411).
- Running the Communication Plug-in and other software clients, see Software Clients (page 409).
- Providing different audio processing based on the audio mode, see Audio (page 413).
- Synchronizing the audio output and input sample clocks, see Synchronization (page 447).

23.2.2.3 User Input Devices

The accessory must provide a touchscreen, or a combination of a rotary knob and at least select and back buttons. See User Input (page 456).

23.2.2.4 Speakers and Microphone

The accessory must provide audio output and input via the vehicle's speakers and a microphone in the vehicle cabin. See Audio (page 413).

23.2.2.5 Siri Button

Siri is an integral part of the CarPlay experience. Siri can be used in conjunction with the display to call people, select and play music, hear and compose text messages, and get directions. Siri can also be used to access Apple device features that do not appear on the display, including notifications, calendar information, reminders and more. Siri is one of the most important ways that users interact with CarPlay so it's critical that the behavior of Siri is consistent with what users are familiar with on Apple devices.

The accessory must have a tactile button for Siri activation. The Siri button cannot be a soft button that appears on a vehicle display. The Siri button is used to activate a Siri session and to continue a Siri session when there is an on-going conversation. The Siri button must be accessible at all times. Immediately after the Apple device has been connected to the accessory, users expect to be able to activate Siri at any time, regardless of what state the vehicle is in and even if CarPlay is not showing on the display. There are some exceptions, such as when the vehicle is performing a safety critical function or if there is already an active voice recognition session using the accessory's built-in system.

The Siri button must be easily accessible. It is expected that the Siri button is located on the steering wheel, although in some exceptional cases the Siri button may be located elsewhere. In many cases the Siri button will be the same button that is used to activate the accessory's built-in voice recognition system. In this case, a short press on the button should trigger the accessory's built-in voice recognition system and a long press on the button should trigger Siri. The accessory is responsible for defining what constitutes a long press on the button, but it must not be greater than 1000 ms. Apple recommends a period of 600 ms since it closely

mimics the behavior of the Apple device home button. Once the user activates Siri by holding down the button, a Siri session is initiated. The accessory knows that a Siri session is active by observing the application state which will be "Speech". While the Siri session is active, the accessory must send all Siri button events (every time the Siri button is either pressed or released) to the Apple device. The Apple device monitors both button press events and button release events and manages the Siri session accordingly. The accessory must send raw events and should not attempt to interpret the button press and button release events. When the Siri session is terminated, the accessory may stop sending button events and return to a default state.

If the Siri button is a dedicated hardware button used exclusively for Siri and is never used for built-in voice recognition activation or for other features using other devices or systems, then you may choose to label the button with Siri branding. In this case, please consult with Apple for guidelines on using the Siri brand. Otherwise Siri should not appear on the button and it should be labeled with standard iconography, such as the icon used for the built-in voice recognition system.

If the accessory supports CarPlay over wireless, the Siri button must also function as a trigger to start the pairing process. When no device is connected, a long press on the Siri button must activate the accessory's wireless pairing user interface and the accessory must immediately become discoverable by an Apple device. However, if the accessory is actively connected to a second Bluetooth device, a long press on the Siri button may be used to trigger voice recognition activation instead of activating the accessory's wireless pairing user interface.

See requestSiri (page 482).

23.2.2.6 "Apple CarPlay" Button

To show the CarPlay UI, the accessory must provide one of the following:

- A physical button.
- A soft button in a top-level menu.
- A menu item in a top-level menu.

The Apple CarPlay icon may be used on physical buttons that show the CarPlay UI. The label "Apple CarPlay" should appear in text below the icon. If the icon is not used, the text-only label "Apple CarPlay" must be used.

The Apple CarPlay logo may be used on soft buttons and menu items that show the CarPlay UI. If color graphics are featured in the accessory screen interface, the accessory should use the color version of the Apple CarPlay logo. If black-and-white graphics or other monochromatic themes are featured in the accessory screen interface, use either the white version or the black version of the Apple CarPlay logo, matching the visual style of the interface. The logo must be accompanied by the label "Apple CarPlay". If the logo is not used, the text-only label "Apple CarPlay" must be used.

Whenever the CarPlay session is not active, the Apple CarPlay logo and/or label should appear disabled or should be hidden.

When using the label "Apple CarPlay", the accessory should match the font, size, style, and placement of other text used in the interface; do not imitate Apple typography. For example, if all capital letters are used on other items, then Apple CarPlay can also appear in all capital letters: "APPLE CARPLAY".

It is preferred that "Apple CarPlay" is placed on one line. If space is limited and other titles are stacked, "Apple CarPlay" can be stacked, with "Apple" on one line and "CarPlay" on the next line.

Any use of the Apple CarPlay icon, logo, or trademark within native UI requires Apple review and approval.

See the *Apple CarPlay Identity Guidelines* for more details.

See requestUI (page 483) to show the CarPlay UI.

23.2.2.7 Sensors

The accessory must provide location information to the Apple device as described in Location Information (page 637).

23.2.2.8 Connection to the Apple device

To connect to the Apple device over USB, the accessory must provide a USB-A receptacle or a Lightning connector, see CarPlay over USB (page 388).

To connect to the Apple device wirelessly, the accessory must provide a Bluetooth connection and a wireless access point, see CarPlay over Wireless (page 395).

Refer to the *Apple CarPlay Identity Guidelines* for labeling any receptacles or connectors.

23.2.2.9 Device Mount Location

The Apple device should be mounted in an open holder that is away from direct sunlight or any heat source. Air ventilation is recommended but not required, however an enclosed compartment with no air circulation, such as the glove box, should be avoided.

23.2.3 Architecture

To support CarPlay, an accessory must support USB as a USB device where the Apple device is the host, see USB Host Mode (page 712).

To support CarPlay over wireless, an accessory must support Bluetooth and Wi-Fi, see CarPlay over Wireless (page 395).

The accessory communicates with the Apple device using protocols defined by the Internet Protocol (IP) suite. An IP model is employed to abstract data transport away from the chosen physical layer where possible, although some aspects of the overall implementation will differ per transport. In addition, the accessory must establish an iAP2 link with the Apple device for authentication, identification, and exchange of metadata.

The major building blocks of this architecture are illustrated in Figure 23-1 (page 387), with Apple-provided software components highlighted in blue.

Figure 23-1 Topology of an automotive head unit using CarPlay

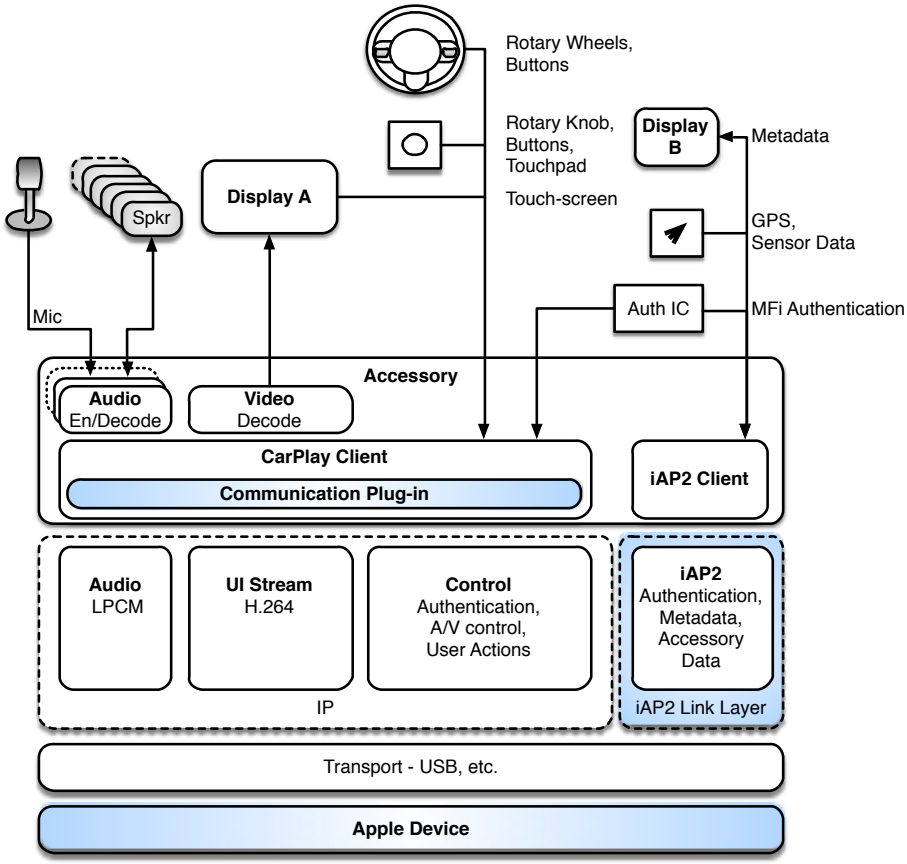


Figure 23-1 (page 387) shows an automotive head unit and its associated instrument cluster and displays with:

- Digital content that is independently driven by an Apple device
- User interactions that are controlled by one or more discrete controllers
- Multichannel audio, both input and output

23.2.4 CarPlay over USB

At a minimum, the accessory must support Hi-Speed USB; see the *USB 2.0 Specification*. Because of the bandwidth and low-latency requirements for transferring interactive audio/visual content, the use of a dedicated USB controller for the interface between the Apple device and accessory is recommended.

The accessory must also support the USB Host Mode feature (where the accessory is a USB device and the Apple device is the host), as specified in USB Host Mode (page 712).

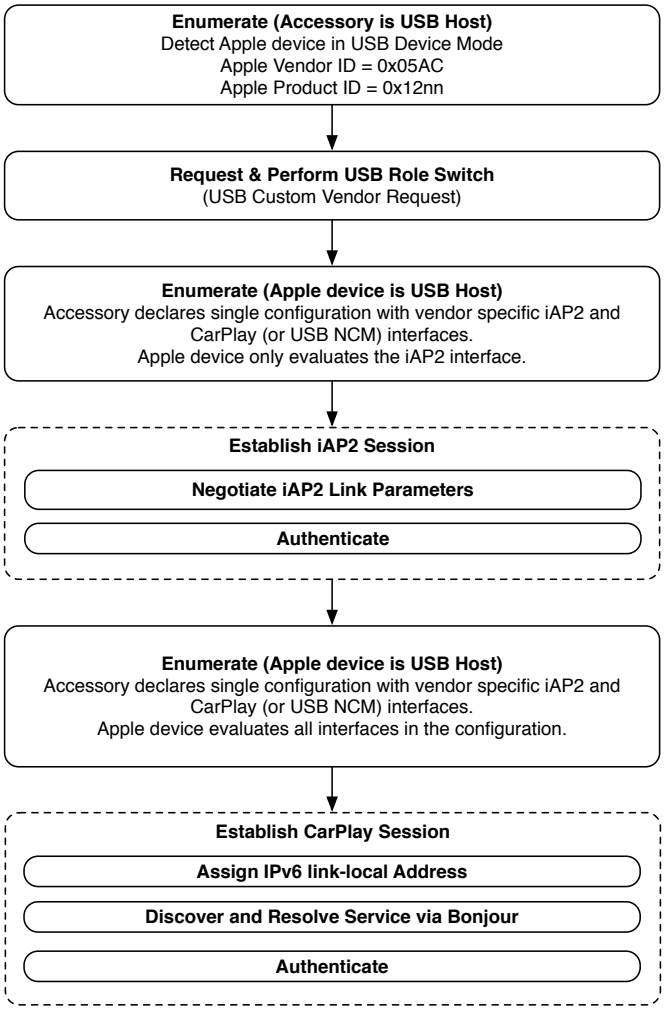
USB Network Control Model (NCM) is defined in the following specifications:

- *Universal Serial Bus Communications Class Subclass Specifications for Network Control Model Devices, Revision 1.0*
- *Universal Serial Bus Class Definitions for Communication Devices, Version 1.2*

The accessory must establish the CarPlay session automatically upon physical connection to the Apple device.

For an accessory that normally acts as a USB host, the steps shown in Figure 23-2 (page 389) are required.

Figure 23-2 Process for establishing a CarPlay session



23.2.4.1 Role Switch

If the accessory normally acts as a USB host, it must perform a host-to-device role switch as specified in USB Role Switch (page 714).

An accessory that presents directly as a USB device without undergoing the host-to-device role switch, will require the use of a custom USB cable or dock solution, incorporating the Lightning (C10A), Lightning (C11A), Lightning (C48A), or Lightning (C68A) connector (USB Host Mode). The standard Apple-provided white USB-to-Lightning cable is incompatible with a USB accessory presenting directly as a USB device.

23.2.4.2 iAP2 / NCM Interface Configuration

Once role switch is completed, the accessory must continue by enumerating the supported USB interfaces. At a minimum the accessory must present a single configuration which includes:

- An iAP2 interface (see USB Host Mode (page 712)).
- A USB NCM control interface (see Table 23-1 (page 390) and Table 23-2 (page 390)).
- A USB NCM data interface (see Table 23-3 (page 391)).

The accessory must proceed to establish an authenticated iAP2 Session as described in Accessory Authentication (page 261).

> **Note:** Until authentication is completed successfully, the Apple device will access the iAP2 interface unless it is using control session version 2 (see Control Session (page 789)). With version 2, the Apple device may access other USB interfaces whether or not authentication is complete.

Table 23-1 USB NCM Control Interface Descriptor

USB Descriptor	Value	Description
Interface Number	0xNN	Must be different from the iAP2 interface and USB NCM data interface numbers. Must match the USBHostTransportCarPlayInterfaceNumber (see Table 59-15 (page 812)).
Interface Class	0x02	USB Communication Interface Class
Interface Subclass	0x0D	Network Control Model
Interface Protocol	0x00	No encapsulated commands / responses
Number of Endpoints	1	1 Interrupt IN (optional): This is typically used to convey changes in link status. Since link is expected to be maintained at all times, we will synthesize link up if there is a read completion via the data interface.

The accessory must also publish the additional descriptors outlined in Table 23-2 (page 390).

Table 23-2 USB NCM Communication Interface Descriptor Requirements

USB Descriptor	Description
Header	CDC Header functional descriptor
Union	CDC Union functional descriptor
Ethernet	CDC Ethernet Networking functional descriptor

USB Descriptor	Description
NCM	NCM functional descriptor

Accessory MAC addresses must be properly assigned by the manufacturer, see http://www.iana.org/assignments/ethernet-numbers/ethernet-numbers.xhtml. The accessory must provide this MAC address for the Apple device to use via the CDC Ethernet Networking functional descriptor. The accessory must not use the same MAC address provided to the Apple device for its own network interface. The accessory may use a different assigned MAC address, or it may modify the MAC address for its own use by setting the locally administered bit of the Organizationally Unique Identifier (OUI) portion of the MAC address. The locally administered bit is the second-least-significant bit of the most significant byte of the address, see http://standards.ieee.org/develop/regauth/tut/macgrp.pdf. This will ensure that there are no collisions between the Apple device and the head unit.

Table 23-3 USB NCM Data Interface Descriptor

USB Descriptor	Value	Description
Interface Number	0xNN	Must be different from the iAP2 interface and USB NCM control interface numbers.
Interface Class	0x0A	USB Data Interface Class
Interface SubClass	0x00	
Interface Protocol	0x01	NCM Data Class
Number of Endpoints	0	(for Alternate Setting 0)
Number of Endpoints	2	(for Alternate Setting 1) 1 Bulk IN; and 1 Bulk OUT

The accessory must implement USB Hi-Speed NCM on this interface. The interface must support transfers of encapsulated datagrams up to 64 KB (i.e., up to 40 1514-byte Ethernet frames) and 16-bit NCM Transfer Blocks (i.e., NTB-16).

Accessories using the USB NCM interface for CarPlay must support at least 100 Mbps of bandwidth with a latency less than 5 ms over both TCP and UDP protocols and a packet loss of less than 1% (as measured with iperf) over UDP.

When the Apple device is connected or disconnected, the accessory must reflect this change in the connection state of the NCM interface. The network stack on the head unit must mark the NCM interface as available only while the Apple device is connected.

An example configuration for an accessory implementing the USB Communications Class NCM Subclass descriptors is as follows:

```
Device Release                                          :   r2.00
USB Spec Release                                        :   v2.00
Serial Number                                          :   ABC-0123456799
Class                                                   :   0xff
(Vendor-specific)
Subclass                                                :   0x00
Protocol                                                :   0x00
Configurations                                          :   1
    Configuration                                       :   1 (Default
Configuration)
        Attributes                                      :   0xc0
(Self-powered)
        Max Power                                       :   0 mA
        Interfaces                                      :   3
            Interface                                   :   0 / 0 (iAP2
Interface)
                Class                                   :   0xff
                Subclass                                :   0xf0
                Protocol                                :   0x00
                Endpoints                               :   2
                    Endpoint                            :   0x81
                        Attributes                      :   Bulk/IN
                        Max Packet Size                 :   512
                    Endpoint                            :   0x1
                        Attributes                      :   Bulk/OUT
                        Max Packet Size                 :   512
            Interface                                   :   1 / 0 (NCM
Communication Interface)
                Class                                   :   0x02
                Subclass                                :   0x0d
                Protocol                                :   0x00
                Endpoints                               :   1
                    Endpoint                            :   0x82
                        Attributes                      :   Interrupt/IN
```

```
                    Max Packet Size                 :   64
                    Interval                        :   1 mframe
                CDC Header Functional Descriptor    :
                    bcdCDC                          :   v1.10
                CDC Union Functional Descriptor     :
                    Control Interface               :   1
                    Subordinate Interface           :   2
                CDC Ethernet Networking Functional Descriptor   :
                    iMacAddress                     :   7
                    Ethernet Statistics             :   0
                    Max Segment Size                :   1514
                    Number MC Filters               :   0
                    Number Power Filters            :   0
                NCM Functional Descriptor           :
                    NCM Version                     :   v1.00
                    Network Capabilities            :   0x3A
            Interface                               :   2 / 0 (NCM
Data Interface Alternate 0)
                Class                               :   0x0a
                Subclass                            :   0x00
                Protocol                            :   0x01
                Endpoints                           :   0
            Interface                               :   2 / 1 (NCM
Data Interface Alternate 1)
                Class                               :   0x0a
                Subclass                            :   0x00
                Protocol                            :   0x01
                Endpoints                           :   2
                    Endpoint                        :   0x83
                        Attributes                  :   Bulk/IN
                        Max Packet Size             :   512
                    Endpoint                        :   0x3
                        Attributes                  :   Bulk/OUT
                        Max Packet Size             :   512
```

23.2.4.3 Authentication

CarPlay is an authenticated solution, requiring the use of an MFi authentication coprocessor (2.0 B or later) obtained through Apple. Apple devices will stream only to authorized accessories.

Communication over both the iAP2 and CarPlay interfaces requires authentication and each interface provides a discrete authentication API. To expedite these dual authentication steps, local caching of the X.509 Certificate provided by the Apple Authentication Coprocessor is permitted.

All authenticated accessories are required to be certified under the Apple MFi program. The CarPlay accessory must successfully pass compliance tests to assure that all digital content from an Apple device will be correctly decoded and displayed and that all electrical requirements described in this specification are met.

23.2.4.4 Networking and Service Discovery

To establish a CarPlay session, the accessory must be networked with the Apple device by using the communication protocols defined by the Internet Protocol (IP) documentation; see *IETF RFC 2460, Internet Protocol, Version 6 (Ipv6) Specification* , December 1998. IP connectivity must be provided by IPv6 link-local addressing and the accessory must support Apple Bonjour zero-configuration networking over this interface; see the *Apple Bonjour Support Site* . The accessory will be identified uniquely by its MAC address and by a TXT record with additional data needed to resolve or use the service.

For more details, see Discovery (page 424).

23.2.4.5 Session Establishment

Once a connection has been established, setup and content transfer will start after the accessory completes authentication over the CarPlay interface.

The accessory must not send a 'Play' command automatically upon connection of a CarPlay enabled device. See Media Library Playback Requirements (page 650).

For more information on setting up a CarPlay session refer to Setup and Control (page 427).

The accessory must be able to establish a CarPlay session within 3 seconds of Apple device connection.

23.2.4.6 Session Termination

The accessory must be able to detect a disconnect and terminate the session within 1 sec of a physical detachment of the Apple device.

When the accessory also supports CarPlay over wireless (see CarPlay over Wireless (page 395)), if the Apple device is disconnected, the accessory must follow the requirements outlined in USB to Wireless (page 409).

23.2.5 CarPlay over Wireless

CarPlay can automatically connect the car wirelessly, without needing to handle or plug in the Apple device - perfect for short drives.

Setup for CarPlay over wireless begins with Bluetooth discovery between the Apple device and the accessory. The user initiates pairing on the accessory either through a long press on the accessory's Siri button, or by using the accessory's user interface. On the Apple device, the user initiates pairing on the Apple device through Settings. The accessory and the Apple device may choose to show all detected Bluetooth devices, or they may choose to show only devices that support CarPlay over wireless by querying the Bluetooth Extended Inquiry Response (EIR). A Bluetooth Classic pairing flow generates a record of trust between the Apple device and the accessory.

After Bluetooth pairing is complete, the Apple device requests the accessory's Wi-Fi credentials using iAP2 over Bluetooth. The Apple device uses the Wi-Fi credentials and the Apple Device Information Element provided by the accessory's access point to associate with the accessory's Wi-Fi network.

Once associated with the network, the Apple device advertises a CarPlay Control Bonjour Service to indicate that CarPlay is available. The accessory is responsible for requesting initiation of the CarPlay session using CarPlay Control. Once the CarPlay session starts, the user can interact with CarPlay through the accessory's controls. Finally, the accessory transitions from iAP2 over Bluetooth to iAP2 over CarPlay client.

Once initial pairing is completed, Apple CarPlay reconnects seamlessly the next time the user enters the vehicle. Reconnection starts with the accessory reestablishing the Bluetooth connection. This is used as a trigger for the Apple device to associate with the already known Wi-Fi network and to start advertising the CarPlay Control Bonjour service. Once the accessory discovers a CarPlay-enabled device though the CarPlay Control Bonjour service, it requests a CarPlay session from that device.

For accessories that support multiple Apple devices with CarPlay, including CarPlay over USB and CarPlay over wireless, the user selects which Apple device to use for CarPlay from a list presented in the accessory's user interface. The accessory can add a CarPlay device to the list by following the same steps to reconnect each previously paired Apple device: a Bluetooth connection is established, the Apple device associates with the Wi-Fi network, the Apple device advertises CarPlay availability through the CarPlay Control Bonjour service. If an Apple device is connected over USB and over wireless, it appears only once in the list.

23.2.5.1 Bluetooth

CarPlay over wireless requires the accessory to provide Bluetooth connection, service discovery, and pairing.

During initial pairing, the accessory must support standard Bluetooth Secure Simple Pairing using Numeric Comparison. Once a secure Bluetooth link is established, the accessory must negotiate the iAP2 profile and establish an iAP2 session for exchanging the Wi-Fi credentials, see iAP2 Client over Bluetooth (page 409).

After starting iAP2, the accessory may negotiate all of the relevant Bluetooth profiles such as HFP, A2DP, AVRCP, etc. However, once a CarPlay session is established, the Apple device will notify the accessory to disconnect all active profiles, see disableBluetooth (page 478).

If CarPlay over wireless is available on the Apple device as indicated by the Bluetooth EIR then the accessory must start iAP2 prior to any additional Bluetooth profiles.

Accessories may use the WirelessCarPlayUpdate (page 842) message to receive notifications about the availability of CarPlay over wireless on an Apple Device. If legacy Bluetooth profiles, such as HFP, A2DP, etc. are supported and not already connected, the accessory must re-establish the connection when CarPlay over wireless is no longer available on the Apple device.

See Bluetooth (page 344) for additional requirements.

23.2.5.1.1 Accessory CarPlay Bluetooth EIR

The Apple device determines whether the accessory supports CarPlay over wireless by examining the CarPlay UUID included in the accessory's Bluetooth EIR. Accessories that support CarPlay over wireless must respond with the CarPlay Service UUID defined below.

Once the accessory is discoverable, it must run periodic inquiry scans and respond to an inquiry from the Apple device with an FHS packet with the BT EIR bit set as defined in the *Core Specification Supplement, Part A*. It must then return an Extended Inquiry Response packet 1250 microseconds after the FHS of the packet.

The EIR packet must include a 128-bit CarPlay Service UUID, `0xEC884348CD4140A29727575D50BF1FD3`, in one of the following EIR data types:

- Incomplete List of 128-bit Service Class UUIDs
- Complete List of 128-bit Service Class UUIDs

The Extended Inquiry Response may contain additional data as defined in the *Core Specification Supplement, Part A*. DM1 or DM3 packets, which support FEC, are recommended for transmitting the EIR data for maximizing the range and increasing the robustness of the packet.

For more information on Bluetooth EIR, see Additional Specifications (page 381).

23.2.5.1.2 Apple Device CarPlay Bluetooth EIR

Apple devices which support CarPlay over wireless and are discoverable via Bluetooth will advertise the following 128-bit UUID: `0x2D8D2466E14D451C88BC7301ABEA291A`, in one of the following EIR data types:

- Incomplete List of 128-bit Service Class UUIDs
- Complete List of 128-bit Service Class UUIDs

Accessories may use this information to distinguish between devices with Apple CarPlay enabled and general Bluetooth devices.

23.2.5.1.3 Accessory iAP2 Bluetooth EIR

The accessory must support iAP2 over Bluetooth, see iAP2 Client over Bluetooth (page 409).

23.2.5.2 Wi-Fi Access Point

CarPlay accessories must operate as a standard Wi-Fi access point allowing Apple devices to join as a standard Wi-Fi client.

During initial pairing, the Apple device requests the Wi-Fi access point's Wi-Fi credentials (SSID and Passphrase) using iAP2 over Bluetooth, see iAP2 Client over Bluetooth (page 409). These Wi-Fi credentials are stored on the Apple device and will be used to join the Wi-Fi access point automatically during reconnection. To speed up the reconnection scenario, the Apple device will initiate Wi-Fi scanning based on a re-connect of any Bluetooth profile.

The CarPlay accessory must ensure that the Wi-Fi access point is fully operational before it initiates pairing or reconnection of the device over Bluetooth.

Once the Apple device joins the Wi-Fi access point, it advertises the CarPlay Control Bonjour service and waits for the accessory to establish the CarPlay session, see Networking and Service Discovery (page 402).

The accessory Wi-Fi access point implementation must transmit a de-authentication packet to the Apple device before the Wi-Fi access point goes offline (accessory is being turned off).

The accessory must not implement any policy that tracks or filters Apple devices based on Wi-Fi MAC addresses.

The Wi-Fi access point must support at least 25 Mbps of bandwidth (as measured with iperf) with a latency less than 16 ms (as measured with the ping command) over both TCP and UDP protocols and a packet loss of no more than 1% on a 25 Mbps UDP uplink stream.

The Wi-Fi access point may support multiple Wi-Fi connected devices for other applications, but those connections must not impact the required CarPlay performance.

The Wi-Fi access point must be configured for a power save operation using a DTIM value of one (1).

The Wi-Fi access point must not use a hidden SSID.

23.2.5.2.1 Hardware Requirements

At a minimum the accessory must support one of the following physical level data rates and modulations as defined in the *IEEE 802.11-2012* standard:

- 802.11n 2.4 GHz HT20
- 802.11n 5 GHz HT20 or HT40

It is recommended that the accessory supports the 802.11ac VHT20, VHT40, or VHT80 physical level data rates and modulations as defined in the *IEEE 802.11ac-2013* standard.

For more information on the IEEE 802.11-2012 and 802.11ac-2013 standards, see Additional Specifications (page 381).

23.2.5.2.2 Frequency Bands

The Wi-Fi access point may operate in either the 2.4 GHz or the 5 GHz frequency bands.

When hosting a wireless network in the 2.4 GHz frequency band, the Wi-Fi access point must operate on one of the channels in Table 23-4 (page 398).

Table 23-4 Operational Channels for 2.4 GHz Wi-Fi

Channel	Frequency
1	2.412 GHz
6	2.437 GHz
11	2.462 GHz

When hosting a wireless network in the 5 GHz frequency band, the Wi-Fi access point must operate on one of the channels in Table 23-5 (page 398).

Table 23-5 Operational Channels for 5 GHz Wi-Fi

Band	Channel	Frequency
UNII-I (Lower Band)	36	5.180 GHz
UNII-I (Lower Band)	40	5.200 GHz
UNII-I (Lower Band)	44	5.220 GHz
UNII-I (Lower Band)	48	5.240 GHz
UNII-III (Upper Band)	149	5.745 GHz
UNII-III (Upper Band)	153	5.765 GHz

Band	Channel	Frequency
UNII-III (Upper Band)	157	5.785 GHz
UNII-III (Upper Band)	161	5.805 GHz

It is highly recommended that the accessory Wi-Fi access point operate in the 5 GHz frequency band. Operating in the 5 GHz frequency band will avoid interference and coexistence with Bluetooth traffic that only exists in the 2.4 GHz frequency band.

Wi-Fi operation in the 5 GHz frequency band depends on regulatory domain rules and regulations. It is understood that some regulatory domains may prohibit operation in the 5 GHz frequency band, therefore the accessory Wi-Fi access point will have to operate in the 2.4 GHz frequency band.

Channel switching should be limited as it increases connection time. Channel switching should be avoided across accessory power cycles and should not change during a CarPlay over wireless session.

23.2.5.2.3 Basic Wi-Fi Requirements

The accessory must support the Software Access Point (SWAP) Wi-Fi Operational Mode.

The accessory must support Distributed Coordination Functions (DCF).

The accessory must support the following frame types:

- Association Request and Response
- Re-association Request and Response
- Probe Request and Response
  - Broadcast Probe Requests
  - Directed Probe Requests
- Beacons
- Disassociation
- De-authentication
- RTS/CTS
- ACK
- Data Frames
- Null Frames

The accessory must support the following frame reception and transmission functionality:

- Reception and Transmission of Data Frames
- Reception and Transmission of Management/Control Frames
- Reception and Transmission of Public Action Frames
- Receive Defragmentation
- Transmit Fragmentation (optional)

The accessory may implement the mandatory tallies and counters defined in the *IEEE 802.11 Management Information Base (MIB)*.

The accessory must support standard power management and power save functions as defined in the IEEE 802.11-2012 standard, see Additional Specifications (page 381).

The accessory may support short guard interval (400 ns) for the defined Data Rates and MCS indices.

The accessory must support at least the following OFDM Data Rates: 6, 9, 12, 18, 24, 36, 48, and 54 Mbps.

23.2.5.2.4 Advanced Wi-Fi Requirements

The accessory must support the WFA Wireless Multimedia (WMM) Quality of Service (QOS) mechanism, see Additional Specifications (page 381).

The CarPlay protocol uses AC_VO for Voice data traffic, AC_VI for Screen data traffic, and AC_VO for Control data traffic.

The accessory may support Universal Advanced Power Save Delivery (U-APSD) as defined in the IEEE 802.11-2012 standard, see Additional Specifications (page 381). It is strongly recommended that implementations support this power save mode in order to optimize power consumption on the Apple device.

When the Apple device transmits a null data packet with the PM Bit set (entering 802.11 power save mode), the accessory must ACK the null data packet and must flush the Tx hardware queue for that wireless client (STA) therefore not transmitting any additional packets to the Apple device.

23.2.5.2.5 Security

The accessory must use the WPA2 Personal security mode as defined by the Wi-Fi Alliance and IEEE.

The accessory may support the mandatory security related tallies/counters defined in the *IEEE 802.11 Management Information Base (MIB)*.

All support encryption algorithms/functions must be executed in hardware unless otherwise described and approved by Apple.

The AES/CCMP encryption may comply with the *Government Security Requirements for Cryptographic Modules FIPS PUB 140-2*.

23.2.5.2.6 Performance

The following table defines recommended throughput performance targets for the different IEEE 802.11 MAC/PHY modes for the TCP and UDP protocols.

Table 23-6 Expected Throughput for IEEE 802.11 MAC/PHY Modes

IEEE 802.11 MAC/PHY Radio Modes	1SS		2SS	
TCP	**UDP**	**TCP**	**UDP**	
802.11n 2.4 GHz - HT20	55	65	100	130
802.11n 5 GHz - HT20	55	65	100	130
802.11n 5 GHz - HT40	120	135	210	270
802.11ac 5 GHz - VHT20	60	70	120	140
802.11ac 5 GHz - VHT40	140	160	280	320
802.11ac 5 GHz - VHT80	300	350	600	700

23.2.5.2.7 Wi-Fi Alliance Compliance and Conformance

The CarPlay accessory Wi-Fi implementation must be in compliance with the mandatory to implement requirements defined in the *Wi-Fi CERTIFIED(tm) 802.11 ac Interoperability Test Plan*.

The CarPlay accessory Wi-Fi implementation must be in compliance with the mandatory to implement requirements defined in the *Wi-Fi 802.11n System Interoperability Test Plan*.

23.2.5.2.8 Interworking Information Element (IE)

The accessory must include the IEEE 802.11 Interworking IE in Beacon, Probe response, and Association response frames at all times during the operation of the Wi-Fi access point.

The accessory must set the following fields:

- "Access Network Options" field - must be set based on the availability of Internet connectivity, see Enabling Internet Data Connectivity (page 403).
- "Venue Info" field - must be set to 10 (Vehicular). The Venue Type code must be set to one of the values that correspond to Venue Group code 10 (0-255) as defined in *IEEE Std. 802.11-2012*.

For more information on the Interworking IE, see Additional Specifications (page 381).

23.2.5.2.9 Apple Device Information Element (IE)

The accessory must support and include the Apple Device IE, see Apple Device Information Element (IE) (page 743).

The accessory must include the Apple Device IE in Beacon, Probe response, and Association response frames at all times during the operation of the Wi-Fi access point.

The Features flags must set the flags as specified in Table 23-7 (page 402) (see Table 55-8 (page 746)):

Table 23-7 Features flag settings

Flag	Value
Supports CarPlay over Wireless	1
Provides Internet access	1 if Internet data connectivity is supported, provisioned, and enabled.
Supports 2.4 GHz Wi-Fi networks	1 if the 2.4 GHz frequency band is supported and used for CarPlay.
Supports 5 GHz Wi-Fi networks	1 if the 5 GHz frequency band is supported and used for CarPlay.

The following parameters are required, see Payload (page 745):

- Name
- Manufacturer
- Model
- OUI
- Bluetooth MAC Address
- Device ID

23.2.5.3 Networking and Service Discovery

To establish a CarPlay over wireless session, the accessory must be networked with the Apple device using the communication protocols defined by Internet Protocol (IP) documentation, see *IETF RFC 791, Internet Protocol (IP), DARPA Internet Program, Protocol Specification* , September 1981.

IP connectivity must be provided by IPv4 DHCP addressing and the accessory must support TCP and UDP transmission modes; see *IETF RFC 2131, Dynamic Host Configuration Protocol (DHCP)* , *IETF RFC 793, Transmission Control Protocol (TCP)* and *IETF RFC 768, User Data Gram Protocol (UDP)* . In addition, the accessory must support the following networking protocols: ARP and ICMP; see *IETF RFC 826, Address Resolution Protocol (ARP)* and *IETF RFC 792, Internet Control Message Protocol (ICMP)* .

The accessory must support Apple Bonjour zero-configuration networking over this interface; see the *Apple Bonjour Support Site* . The accessory will be identified uniquely by its MAC address and by a TXT record with additional data needed to resolve or use the service. For more details, see Discovery (page 424).

In order to reduce reconnection time, the accessory must store the DHCP IP address lease information across multiple reboots. The DHCP IP address lease time must be at least 3 days with an address pool size of at least 100; a lease time of 7 days with a pool size of 250 is recommended.

23.2.5.4 Enabling Internet Data Connectivity

Internet connection is defined as an active data connection to servers that reside on the Internet. In case of a cellular based Internet connection the SIM must be valid, provisioned, and enabled for data. The accessory must indicate if it provides an active Internet connection that can route data.

Based on availability of Internet data connection, the accessory must update the following fields:

- "Access Network Options" flag in the Interworking IE, see Interworking Information Element (IE) (page 401)
- "Provides Internet Access" flag in the Apple device IE, see Apple Device Information Element (IE) (page 402)
- DHCP server gateway IP address - a valid DHCP server gateway IP address if Internet connectivity is provided, otherwise the gateway IP address must be set to 0.0.0.0

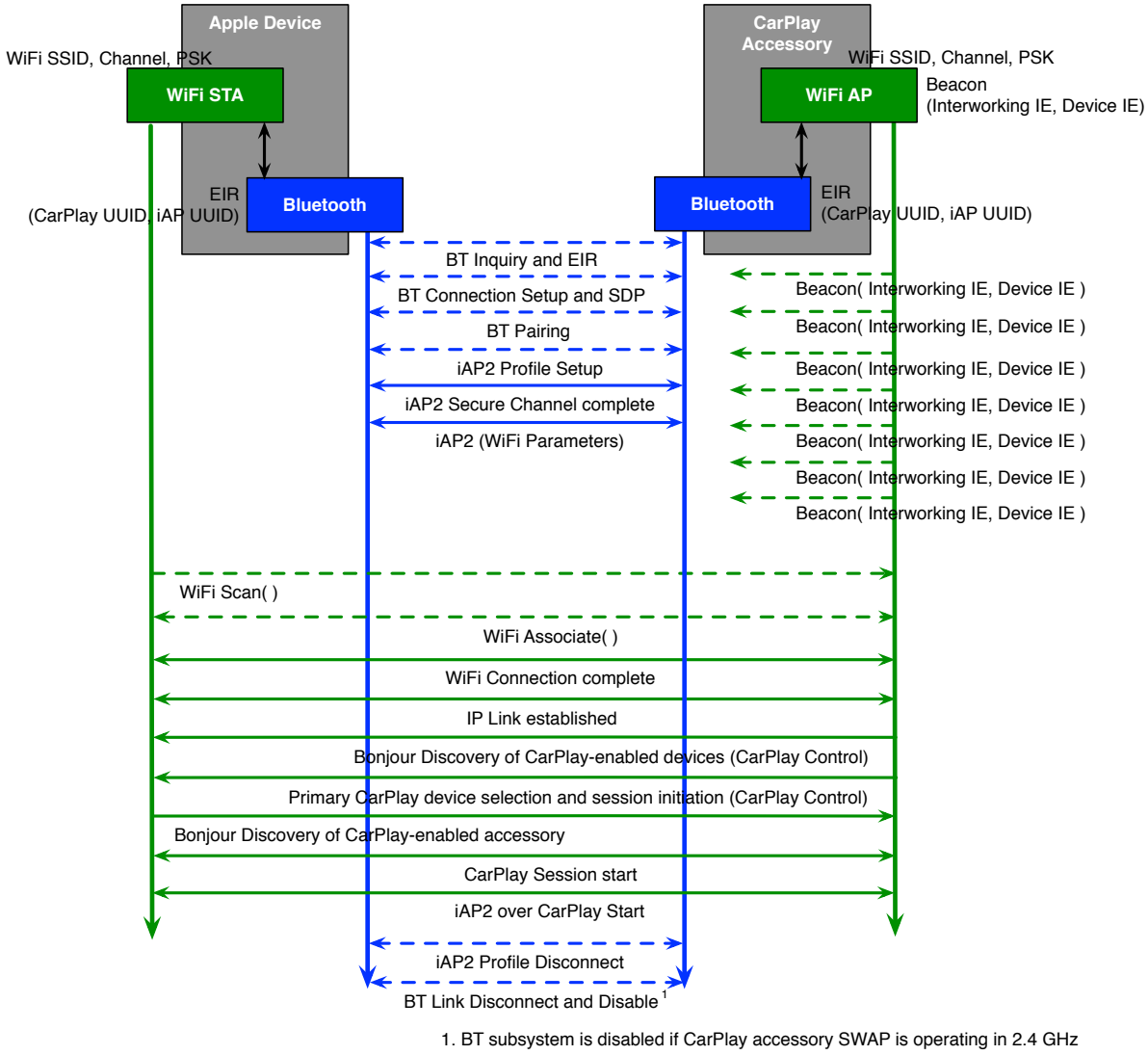
23.2.5.5 Session Establishment

Once the Apple device has successfully received the Wi-Fi credentials over iAP2, the Apple device will scan for the accessory Wi-Fi access point and associate using the provided parameters. Once the Wi-Fi connection is complete, IP will be brought up, Bonjour discovery and device selections occurs, and the CarPlay session can be initiated as defined in Discovery (page 424).

Once the CarPlay session is successfully initiated, if the accessory Wi-Fi access point is operating in 2.4 GHz, Bluetooth link with the Apple device must be disconnected and the Bluetooth subsystem must be disabled (see disableBluetooth (page 478)). If the accessory Wi-Fi access point is operating in 5 GHz, Bluetooth link with the Apple device must be disconnected and the Bluetooth subsystem must go into idle mode, available for connections with other devices.

Figure 23-3 (page 404) defines the message flow when using Bluetooth and iAP2 over Bluetooth as the initial connection and pairing mechanism.

Figure 23-3 Initial Connection and Pairing using Bluetooth and iAP2



23.2.5.6 Session Reconnection

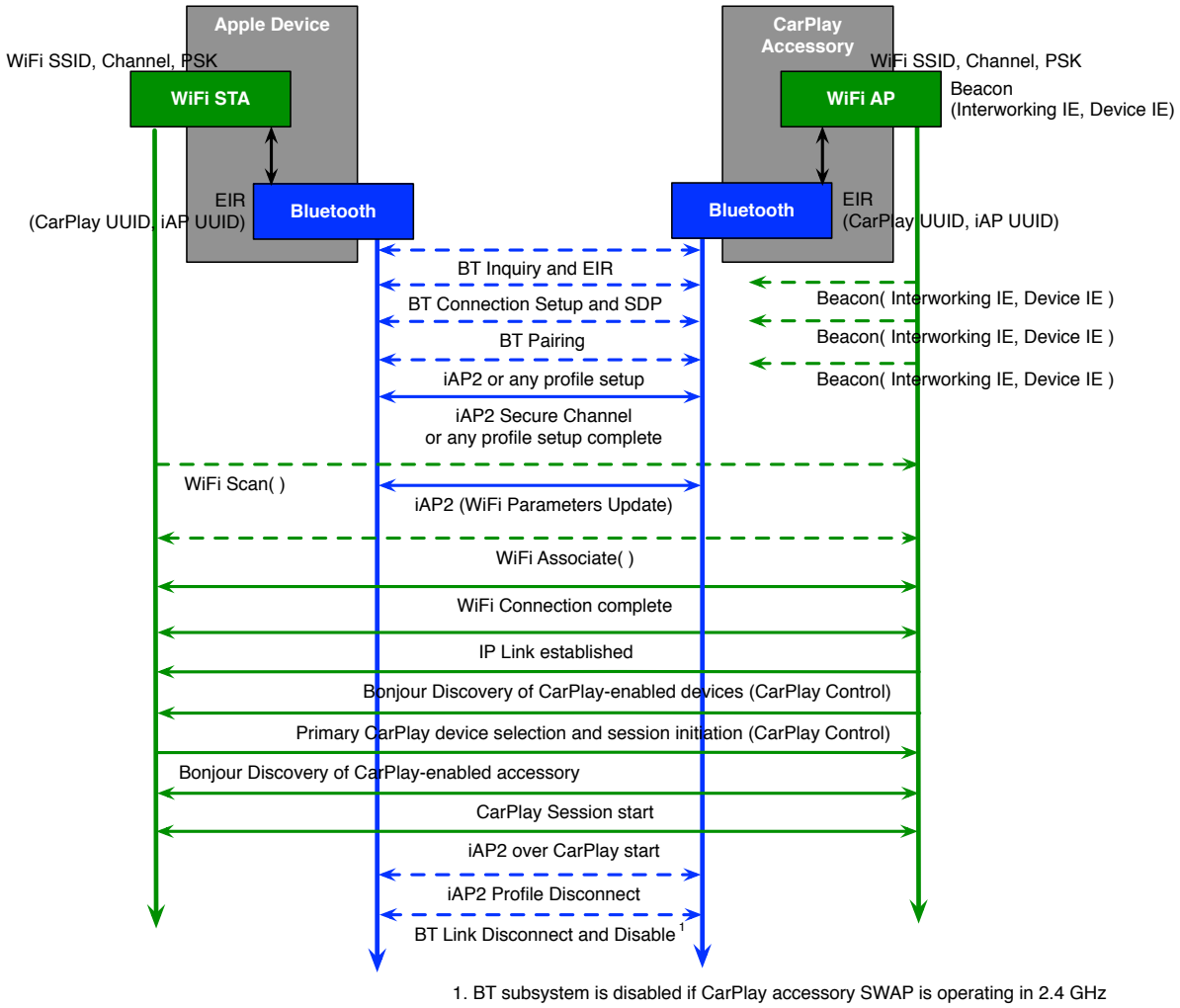
This section defines the reconnection procedure and requirements when using Bluetooth.

The accessory must initiate reconnection to a previously paired Apple device using Bluetooth. The accessory Wi-Fi access point must be fully operational before initiating reconnection. Once the Bluetooth link is established, the Apple device will scan and attempt to associate to the accessory's Wi-Fi access point using the Wi-Fi credentials mapped to the BD_ADDR of the accessory in question. Just like with the Initial Connection and Pairing procedure, the accessory must exchange the Wi-Fi credentials using iAP2 over Bluetooth.

Once the CarPlay session is successfully initiated, if the accessory Wi-Fi access point is operating in 2.4 GHz, Bluetooth link with the Apple device must be disconnected and the Bluetooth subsystem must be disabled (see disableBluetooth (page 478)). If the accessory Wi-Fi access point is operating in 5 GHz, Bluetooth link with the Apple device must be disconnected and the Bluetooth subsystem must go into idle mode, available for connections with other devices.

Figure 23-4 (page 405) defines the message flow when using Bluetooth for reconnection.

Figure 23-4 Reconnect using Bluetooth



For discovery and connection establishment, the reconnect latency must be 8 seconds or less. Reconnect latency is defined as the period from the when the accessory's Bluetooth subsystem transmits the first Bluetooth connection packet (Bluetooth Connection and SDP Setup) to the Apple device to when the Apple device starts the CarPlay session (CarPlay Session Start).

In the event the Apple device with the active CarPlay session is disconnected, the accessory must enable the Bluetooth subsystem and attempt to reconnect to the Apple device. This requirement is meant to re-establish the CarPlay session with an Apple device when a sudden Wi-Fi disconnect occurs.

23.2.5.7 Supporting Multiple Devices

At any time, there is only one active CarPlay session. However, accessories operating in the 5 GHz frequency band may support multiple Apple devices connected to the accessory's Wi-Fi access point by providing a user interface to select which Apple device to actively use for CarPlay. If the accessory provides a user interface to select the active CarPlay device, it must display a list of available Apple devices obtained by actively scanning for all previously paired Bluetooth devices, and using the CarPlay Control Bonjour Service to obtain the status of each Apple device.

Upon ignition, the accessory must scan for all previously paired Apple devices. When Apple devices are discovered, the accessory must attempt to reconnect to the last device used for CarPlay, or to the preferred device that is set up by the user. If neither is found, the accessory must connect an available Apple device.

When there is an active CarPlay session, the accessory must never switch to another Apple device without explicit user action. If the active CarPlay device is disconnected for any reason, the accessory must rescan for available devices and attempt to reconnect the previously active CarPlay device.

Accessories operating in the 2.4 GHz band must support only a single Apple device for CarPlay over wireless. The accessory must not allow Bluetooth pairing while there is an active CarPlay session.

23.2.5.8 Wireless Coexistence

Accessories incorporating other RF technologies such as Bluetooth, multiple Wi-Fi access points, LTE, wireless audio systems, etc. must first consult with Apple to plan co-existence scenarios and testing.

23.2.5.8.1 Wi-Fi and Bluetooth Coexistence

CarPlay over wireless Wi-Fi and Bluetooth protocol traffic may interfere with each other when operating in the 2.4 GHz frequency band.

If the accessory Wi-Fi access point is operating in the 2.4 GHz frequency band, the Bluetooth subsystem must be disabled. Once the Apple device has established a Wi-Fi connection and successfully initiated a CarPlay session, the Bluetooth link must be terminated with the Apple device, the accessory's Bluetooth subsystem must be disabled, and all Bluetooth connections to other devices must be terminated.

If the accessory Wi-Fi access point is operating in the 5 GHz frequency band, Bluetooth protocols and profiles can be supported and enabled with other devices that are not engaged in an active CarPlay session. Once the Apple device has established a Wi-Fi connection and successfully initiated a CarPlay session, the Bluetooth link must be terminated with the Apple device. A Bluetooth connection and profiles may be established with other devices that are not engaged in a CarPlay session.

23.2.5.8.2 Wi-Fi and Cellular Coexistence

CarPlay over wireless and Internet Sharing Services using LTE on Band 40 may interfere with each other when operating in the 2.4 GHz frequency band. In order to maintain the required performance and user experience, the accessory Wi-Fi access point will be limited to a set of 2.4 GHz operating channels.

If the accessory provides Internet sharing services and uses LTE on Band 40 as the WWAN communication, then the accessory Wi-Fi access point must use the operating channels in Table 23-8 (page 407).

Table 23-8 Operational Channels for Cellular Coexistence

Channel	Frequency
6	2.437 GHz
11	2.462 GHz

If the accessory can demonstrate a minimum of 30 dBm of isolation between the Wi-Fi antenna and Cellular antenna, then the accessory Wi-Fi access point may use any 2.4 GHz channels specified in Table 23-4 (page 398).

23.2.5.8.3 Wi-Fi and Coexistence with Other RF Technologies

If the accessory supports other RF technologies in the vehicle, the manufacturer must communicate to Apple the operating properties of these RF technologies:

- Frequency Band
- Channel Width
- Protocol

Other RF technologies operating in the vehicle must not interfere with the Wi-Fi access point providing CarPlay over wireless services since they may cause a performance degradation.

23.2.5.8.4 Multiple Wi-Fi Access Points

If the accessory supports multiple Wi-Fi access point in the vehicle, the manufacturer must communicate to Apple the operating parameters of other Wi-Fi access points that are not intended for CarPlay over wireless.

- Frequency Band
- Channel Width
- Wi-Fi Protocol
- Security Mode
- SSID

Other Wi-Fi access points must not interfere with the Wi-Fi access point providing CarPlay over wireless service since they may cause a performance degradation.

If other Wi-Fi access point(s) are configured with a different SSID and WPA2 passphrase, the non-CarPlay Wi-Fi access points must operate in a different channel.

The non-CarPlay Wi-Fi access points must operate in 2.4 GHz and 5 GHz channels that are not used by the CarPlay Wi-Fi access point.

23.2.6 Transitioning Between Wireless and USB

This section defines requirements for accessories supporting CarPlay over both wired and wireless transports.

23.2.6.1 Wireless to USB

Upon detection of a USB connection with an Apple device, the accessory must use the Get Supported Capabilities USB Vendor Request to determine if CarPlay is enabled. If CarPlay is enabled, the accessory must determine if there is already an active CarPlay session over wireless using Device Notifications Usage (page 526) and DeviceUUIDUpdate (page 842).

If the device has CarPlay enabled and there is no active CarPlay session over wireless, the accessory must initiate USB role switch to start CarPlay. See Role Switch (page 389).

If a CarPlay session over wireless is already established with the Apple device, the accessory must perform USB role switch and initiate an iAP2 session for charging only. The accessory must complete accessory authentication and declare their capability to provide power, see Providing Power to the Apple Device (page 661).

If a CarPlay session over wireless is already established with another Apple device, or if CarPlay is not enabled on the device, the accessory may connect to the device for music browsing.

23.2.6.2 USB to Wireless

When any Apple device is disconnected from the accessory via USB, the accessory must initiate Bluetooth discovery, see Bluetooth (page 395). This ensures that the Apple device can be used wirelessly.

23.2.7 Software Clients

The accessory must implement iAP2 and CarPlay clients to support the message protocols and states specific to each interface.

Portions of both clients are provided as reference source code by Apple, namely:

- An iAP2 link layer
- A CarPlay Communication Plug-in, see CarPlay Communication Plug-in (page 485)

23.2.7.1 iAP2 Client over USB

For accessories which support CarPlay over USB, at a minimum, the iAP2 client must support:

- Accessory Authentication (page 261)
- Accessory Identification (page 265) with VehicleInformationComponent (and WirelessCarPlayTransportComponent if the accessory supports CarPlay over wireless)
- Power (page 660)
- Location Information (page 637) and provide location information from GPS and other sensor data

The location information will be used to augment the Apple device's built-in location services and help conserve device power.

Additionally, the iAP2 interface enables transfer of metadata and state information to the accessory from select iOS apps running in CarPlay, such as Music and Telephony. This information may be used to complement the CarPlay experience on an additional accessory screen that is not the primary CarPlay screen; for instance, to display call information on a vehicle instrument cluster display.

Accessories may also provide sensor data and other vehicle status to the Apple device, see Vehicle Status (page 718).

23.2.7.2 iAP2 Client over Bluetooth

For accessories which support CarPlay over wireless, at a minimum, the iAP2 client must support:

- Accessory Authentication (page 261)
- Accessory Identification (page 265) with BluetoothTransportComponent, VehicleInformationComponent, and WirelessCarPlayTransportComponent
- RequestAccessoryWiFiConfigurationInformation (page 877) and AccessoryWiFiConfigurationInformation (page 877)

The iAP2 client over Bluetooth is used only to provide Wi-Fi credentials and must be disconnected once the CarPlay session is established.

23.2.7.3 CarPlay Client

The CarPlay client must implement the logic described in Resource Management (page 448) for resource sharing between the accessory and Apple device. It must also manage user events and the transfer of digital UI and audio between the accessory and the Apple device.

Payload data for the CarPlay client may be encoded according to a variety of standards dependent on the content type, as described in Media Types and Formats (page 411), but the underlying transport layer employed by the Communication Plug-in will consist of one or more channels implementing either Transmission Control Protocol (TCP) or User Datagram Protocol (UDP). For TCP, see *IETF RFC 793, Transmission Control Protocol*, September 1981. For UDP, see *IETF RFC 768, User Datagram Protocol*, August 1980.

The Communication Plug-in will open a number of additional UDP and TCP channels in parallel to handle synchronization, retransmission requests, usage reports for user actions and other custom commands and controls.

The accessory must provide a networking stack that supports multiple concurrent UDP and TCP channels for:

- audio/video transfer
- command and control
- clock synchronization and retransmission

23.2.7.4 iAP2 Client over CarPlay Client

For accessories which support CarPlay over wireless, the CarPlay Communication Plug-in enables iAP2 over CarPlay, see iAPSendMessage (page 484).

At a minimum, the iAP2 over CarPlay client must support:

- Accessory Authentication (page 261)
- Accessory Identification (page 265) with VehicleInformationComponent, LocationInformationComponent, and WirelessCarPlayTransportComponent

- Location Information (page 637) and provide location information from GPS and other sensor data

For CarPlay over wireless, the accessory must provide location information using GNSS Mode, see Global Navigation Satellite System (GNSS) Mode (page 638), as the Apple device may be positioned in a location without any satellite visibility.

Additionally, the iAP2 interface enables transfer of metadata and state information to the accessory from select iOS apps running in CarPlay, such as Music and Telephony. This information may be used to complement the CarPlay experience on an additional accessory screen that is not the primary CarPlay screen; for instance, to display call information on a vehicle instrument cluster display.

Accessories may also provide sensor data and other vehicle status to the Apple device, see Vehicle Status (page 718).

23.2.8 Media Types and Formats

Video content (a User Interface stream) will stream from the Apple device on a dedicated channel. Two additional dedicated channels are required for audio: main audio (input and output) and alternate audio.

All control and video data packets are encrypted (AES-128). The task of encryption/decryption is abstracted for the accessory maker within the Apple-provided Communication Plug-in code.

Apple devices will not stream content subject to DRM (Digital Rights Management) over CarPlay.

23.2.8.1 User Interface (UI) Stream

The UI is streamed from the Apple device as individual, H.264 NAL units configured with AVCC header data and an additional presentation timestamp for scheduling and synchronization.

The AVCC Format is specified in *ISO/IEC STANDARD 14496-15 (AVC file format)* section 5.2.4.1.1. If the accessory decoder does not natively support AVCC format, but instead requires Annex-B format, a set of utility functions are provided in the Communication Plug-in to assist with format conversion. Note however, that this conversion, if required, is software based, not hardware accelerated, and may impact performance and latency.

The H.264 Sequence Parameter Set (SPS) and Picture Parameter Set (PPS) are only sent once, at the beginning of the stream.

In order to minimize video encoding and decoding latency, no frame reordering is used (i.e. no B-frames). The accessory must support the SPS Video Usability Information (VUI) header with the bitstream_restriction_flag set, max_num_reorder_frames of 0, and max_dec_frame_buffering set to the maximum Decoded Picture Buffer (DPB) size.

Video is encoded using full range Y'CbCr, black is 0, white is 255. The accessory must support the SPS video_full_range_flag to reflect this. See Annex E of the *H.264* specification for more information.

The accessory's target resolution, max frame rate, and physical dimensions for the CarPlay content must be reported via the Info Message (page 430). Resolutions natively supported by the Apple device and its set of CarPlay compatible applications are listed in Table 23-9 (page 412).

The accessory must implement an H.264 decoder capable of supporting at least High Profile 3.1 and content with a 800 x 480 resolution at 30 fps using YCbCr 4:2:0 chroma subsampling, but higher frame rates are strongly recommended. The decoder must be capable of supporting the H.264 level corresponding to the resolution and frame rate reported to CarPlay. See Table 23-10 (page 412).

Table 23-9 CarPlay H.264 Video Stream Standard Resolutions

Aspect Ratio	Pixel Dimensions	Frame Rate
16:9	960 x 540, 1280 x 720	60 fps (recommended), 30 fps
15:9	800 x 480	60 fps (recommended), 30 fps
24:9	1920 x 720	60 fps (recommended), 30 fps

Table 23-10 CarPlay H.264 Levels

H.264 Level	Max Bit Rate Wired	Max Bit Rate Wireless	Examples
High Profile 3.1	17.5 Mbps	10 Mbps	800x480@30fps 800x480@60fps 960x540@30fps 1280x720@30fps
High Profile 3.2	25 Mbps	10 Mbps	960x540@60fps 1280x720@60fps
High Profile 4.0	25 Mbps	10 Mbps	1920x720@30fps
High Profile 4.2	25 Mbps	10 Mbps	1920x720@60fps

The accessory's digital video display must natively support one of the standard resolutions encoded by the Apple device and display the CarPlay content in a full screen configuration. The accessory may optionally use a larger display with a resolution higher than one of the standard resolutions and display the CarPlay content in a windowed configuration. Windowed configurations may have additional design requirements and must be reviewed by Apple. The CarPlay content must always be rendered pixel for pixel without resorting to scaling or stretching. See High Resolution Display (page 383) for display requirements.

Accessory makers wishing to offer different target resolutions for the UI stream and/or nonsquare pixel displays must first consult with Apple for compatibility.

The CarPlay UI must appear on the accessory display within 500 ms of either:

- The accessory receiving a screen stream setup request.
- The accessory receiving a session configuration that includes a screen stream.

23.2.8.2 Audio

The accessory must support audio input and output streams encoded in the form of uncompressed LPCM. Volume control and display of volume control UI are the responsibility of the accessory because the Apple device provides only unattenuated line-level audio output.

See Volume Management (page 423) for specific requirements and recommendations.

The accessory must support one main (input and output) and one alternate audio stream from the Apple device as described in Resources (page 448). Alternate audio must be mixed with main audio regardless of whether the source of main audio is the Apple device (for example, the iOS Music app) or the accessory (for example, the FM tuner on a vehicle head unit). In some use cases output on alternate audio might be accompanied by a request to duck the main audio channel (see Ducking (page 423)).

The audio sample rates supported by the accessory must be reported using the Info message, see Info Message (page 430).

Accessories that implement the CarPlay feature must respect the requirements from Multiple Audio Connections (page 528). In addition, they must use the CarPlay audio stream when using the feature and cannot use USB Host Mode Audio Transport in parallel.

The Apple device will at times simultaneously employ different sample rates for the main (input and output) and alternate audio streams, or may require sample rate transitions, for instance when transitioning main audio from music to a phone call. Main and alternate audio must be capable of maintaining independent sample rates. For example, if the main audio channel is operating at 16 kHz for telephony, alternate audio should continue to operate at a higher sample rate, e.g. 44.1 kHz, to preserve the fidelity of alerts and other alternate audio that would mix with phone call audio. All sample rate transitions will be communicated to the accessory by the Apple device using the Setup Message (page 440).

CarPlay over USB uses LPCM for audio. CarPlay over wireless uses raw AAC-LC for high latency audio (Main High Audio) and either OPUS or raw AAC-ELD for low latency audio (Main Audio except "media" and Alt Audio). Accessories supporting CarPlay over wireless must support multiple decode instances and concurrent decode/encode instances.

Also, the Apple device will communicate the intended use of the audio stream to allow the accessory to: (i) adjust its microphone audio processing (such as noise reduction, echo cancellation, frequency response, and gain), and (ii) suppress or eliminate additional noise sources (for example, temporarily turning down the defroster in a vehicle to reduce ambient noise).

After the Apple device sends a Setup Message (page 440) to request main (input and output) or alternate audio streams, the accessory must reply to the setup message within 100 ms and only after:

- It is ready to begin receiving and outputting audio samples (if main audio with input and output was requested). For output, the accessory must output all samples given to it after an acknowledgement of the setup message. For input the accessory must be ready to send audio samples to the Apple device after the acknowledgment of the setup message.
- It is ready to begin to continuously receive audio from the Apple device (if main or alternate audio output was requested). For output, the accessory must output all samples given to it after an acknowledgement of the setup message.

The accessory must support full-duplex audio on the main audio stream at the same sample rate in both input (microphone) and output directions. In addition, input and output must be driven from the same clock to avoid the need for asynchronous sample rate conversion.

The accessory must not buffer more than 10 ms of audio at a time to ensure that transfer delays do not cause gaps in the audio stream.

At no time during interaction with the Apple device must any popping, clicking, or other audible artifacts occur on the accessory, whether it be during transitions between shared resource ownership or mode changes, starting and stopping IO, initiation or cessation of ducking, or any other circumstance. Additionally, volume changes and ducking must be free any zippering or other audible artifacts. The output to the DAC for an unencoded LPCM audio signal sent from the Apple device should be bit-identical to that of an audio signal sent from other sources, assuming those sources allow direct digital transfer.

23.2.8.2.1 Main Audio - Entertainment

For wired sessions, entertainment audio is sent via the low latency stream (Main Audio). For wireless sessions, entertainment audio is sent via the high latency stream (Main High Audio).

Sample Rate

Table 23-11 (page 415) details the audio stream requirements. The accessory must support both 44.1 kHz 16-bit and 48 kHz 16-bit audio. At least one of these two sample rates must be natively supported, that is, a sample rate at which the accessory can perform its audio processing without sample rate conversion. See Table 23-28 (page 435) for a list of audio configurations.

Table 23-11 Entertainment Stream Requirements

Sample Rate	Bit Depth	Channels	Duplexing	Requirement
44.1 kHz	16 bit	stereo	n/a	Required
48 kHz	16 bit	stereo	n/a	Required

Latency

Latency from the receipt of a buffer of LPCM audio by the accessory from the CarPlay Communication Plug-in to the time it is emitted by the accessory's speaker must not exceed 35 ms.

23.2.8.2.2 Main Audio - Telephony

Unless otherwise specified, the accessory must meet the recommendations and test criteria in International Telecommunication Union Recommendations *ITU-T P.1100: Narrowband hands-free communication in motor vehicles* and *ITU-T P.1110: Wideband hands-free communication in motor vehicles*, whichever applies to the current use case.

Proof of compliance with the *ITU-T P.1100* and *ITU-T P.1110* specifications must be submitted to Apple as part of the self-certification audit process.

Sample Rate

Table 23-12 (page 415) details the audio stream requirements. The accessory must accept and provide the required sample rates with equal or higher bit depth.

Table 23-12 Telephony Audio Stream Requirements

Sample Rate	Bit Depth	Channels	Duplexing	Requirement
8 kHz	16 bit	mono	full-duplex	Required (wired only)
16 kHz	16 bit	mono	full-duplex	Required
32 kHz	16 bit	mono	full-duplex	Recommended

Apple recommends implementing support for 32 kHz sampling. This is a key feature of VoLTE. Future voice communication standards may incorporate 48 kHz sample rates and stereo.

Audio Processing

The accessory must perform echo cancellation and noise processing tuned for telephony and provide full-duplex audio that is free of echoes, noise, or other artifacts.

Uplink Output Level

In the nominal test arrangement, the accessory must provide a constant average uplink output level to the device. The use of automatic gain control (AGC) targeting the specified level is recommended. The nominal level measured at the uplink output of the accessory must be A-weighted -30 dB ±2 dB root-mean-square (RMS), expressed in units relative to full-scale (dBFS(A)). Alternatively, the nominal level may be 13 dB ±2 dB SLR if using the ITU measurement procedure.

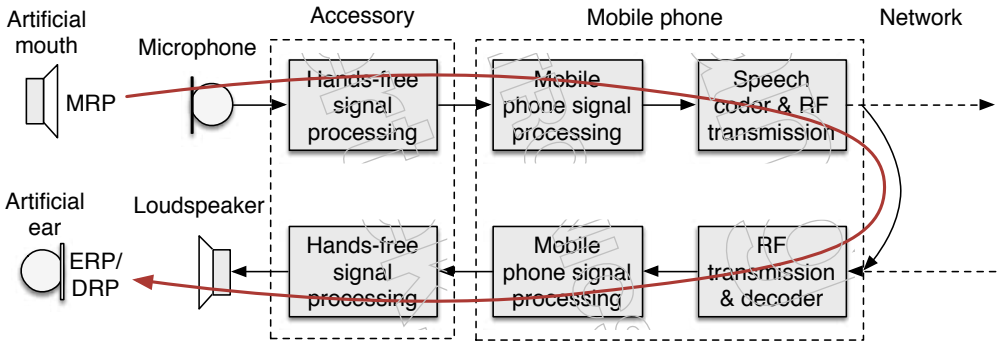
Signal-to-Noise Ratio

The accessory must maintain adequate signal-to-noise ratio (SNR) in its uplink output for voice communication purposes. The absolute SNR must be over 12 dB in all typical driving conditions as specified in Annex D of *ITU-T P.1100* /*ITU-T P.1110* . Higher values are preferred and the SNR should be much higher in a stationary vehicle and at slow driving speeds.

Round-Trip Delay

The definition of the combined round-trip delay of the accessory and the device follows *3GPP TS 26.132* , consisting of the sum of uplink and downlink processing latency of all acoustic, analog, and digital components of the accessory and the device, including any digital buffering. The round-trip path is illustrated in Figure 23-5 (page 416).

Figure 23-5 Telephony Audio Round-Trip Path



The combined round-trip delay of the accessory and the device must not exceed 275 ms for wired transports and 415 ms for wireless transports. Smaller values are preferred and ideally the delay should be under 200 ms.

Send Frequency Response

The send frequency response must meet the requirements in *ITU-T P.1100* , Clause 11.4.1 for narrowband telephony and *ITU-T P.1110* , Clause 11.4.1 for wideband telephony.

23.2.8.2.3 Main Audio - FaceTime Audio

Unless otherwise specified, the accessory must meet the recommendations and test criteria specified in *ITU-T P.1110*.

FaceTime Audio operates at 24 kHz and uses high-quality audio codecs.

Sample Rate

Table 23-13 (page 417) details the audio stream requirements. The accessory must accept and provide the following sample rate with equal or higher bit depth.

Table 23-13 FaceTime Audio Stream Requirements

Sample Rate	Bit Depth	Channels	Duplexing	Requirement
24 kHz	16 bit	mono	full-duplex	Required

In the future, even higher sample rates such as 32 kHz and 48 kHz and stereo may be used.

Audio Processing

The accessory must perform echo cancellation and noise processing tuned for telephony and provide full-duplex audio that is free of echoes, noise, or other artifacts. Due to high-quality audio coding, it is important to preserve the quality of the uplink signal. Less aggressive noise suppression than in telephony should be used if it results in better speech quality in the send direction (see Clause 12.5.1 in *ITU-T P.1110*).

Uplink Output Level

In the nominal test arrangement, the accessory must provide a constant average uplink output level to the device. The use of an AGC targeting the specified level is recommended, but the device should preserve linear operation for any constant speech level. The nominal level measured at the uplink output of the accessory must be A-weighted -30 dB ±2 dB root-mean-square (RMS), expressed in units relative to full-scale (dBFS(A)). Alternatively, the nominal level may be 13 dB ±2 dB SLR if using the ITU measurement procedure.

Signal-to-Noise Ratio

The accessory must maintain adequate signal-to-noise ratio (SNR) in its uplink output for voice communication purposes. The absolute SNR must be over 12 dB in all typical driving conditions as specified in Annex D of *ITU-T P.1110*. Higher values are preferred and the SNR should be much higher in a stationary vehicle and at slow driving speeds.

Round-Trip Delay

The round-trip delay must meet the same requirements as defined for Telephony (see Main Audio - Telephony (page 415)).

Send Frequency Response

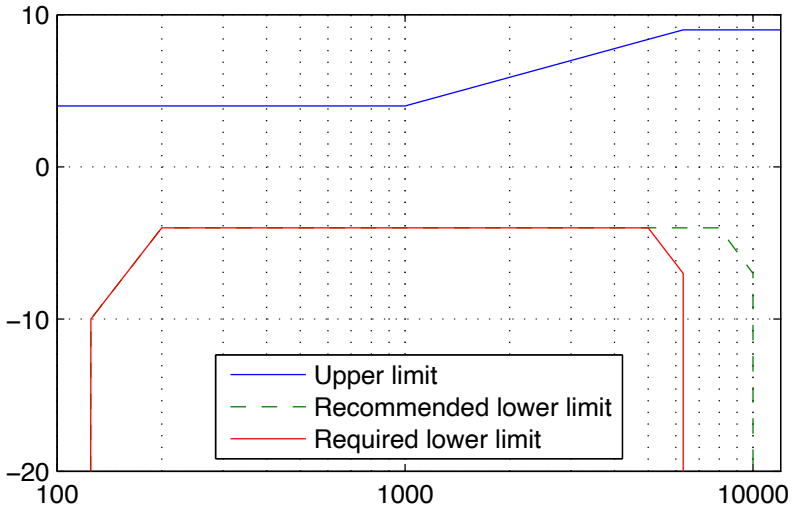
The normalized send frequency response is measured from the mouth reference point (MRP) to the uplink output of the accessory (input to the device). The normalized send frequency response must be within the limits of the tolerance mask shown in Table 23-14 (page 418). The mask (see Figure 23-6 (page 419)) is drawn by straight lines between the breaking points in the table on a linear (dB sensitivity) - logarithmic (frequency) scale. Ideally, the response characteristics of the accessory should be flat in the frequency range of 100 Hz - 11 kHz. Extending the audio bandwidth to above 8 kHz is strongly recommended.

Table 23-14 Audio Tolerance Mask Limits For Send Frequency Response

Frequency	Upper Limit	Recommended Lower Limit	Required Lower Limit
100 Hz	4 dB	-∞ dB	-∞ dB
125 Hz	4 dB	-10 dB	-10 dB
200 Hz	4 dB	-4 dB	-4 dB
1,000 Hz	4 dB	-4 dB	-4 dB
5,000 Hz	(* See note.)	-4 dB	-4 dB
6,300 Hz	9 dB	-4 dB	-7 dB
8,000 Hz	9 dB	-4 dB	-∞ dB
10,000 Hz	9 dB	-7 dB	-∞ dB
12,000 Hz	9 dB	-∞ dB	-∞ dB

* Note: The limits for intermediate frequencies lie on a straight line drawn between the given values on a linear (dB) - logarithmic (Hz) scale. See Figure 23-6 (page 419)

Figure 23-6 Audio Tolerance Mask For Send Frequency Response (Sensitivity (dB) vs. Frequency (Hz))



23.2.8.2.4 Main Audio - Siri

Unless otherwise specified, the accessory must meet the recommendations and test criteria specified in *ITU-T P.1110*.

Sample Rate

Table 23-15 (page 419) details the audio stream requirements. The accessory must accept and provide the following sample rate with equal or higher bit depth.

Table 23-15 Siri Stream Requirements

Sample Rate	Bit Depth	Channels	Duplexing	Requirement
24 kHz	16 bit	mono	full-duplex	Required

Audio Processing

Directional microphones and linear beamforming with microphone arrays giving improved SNR are recommended. Linear echo cancellation for reducing unwanted audio sources (such as audio output from the system) without having any other effect on the speech signal are also recommended. However, single channel noise reduction methods (such as spectrum subtraction) must not be applied, as they will be detrimental to the speech recognition accuracy. Similarly, automatic gain control, residual echo suppression and attempts to blank out non-speech periods in the waveform must not be applied.

Uplink Output Level

In the nominal test arrangement, the accessory must provide a constant uplink output level to the device. The use of an AGC is not recommended. If the accessory adjusts signal gain, the gain must be held constant across each spoken utterance. The nominal level measured at the uplink output of the accessory must be A-weighted -30 dB ±2 dB root-mean-square (RMS), expressed in units relative to full-scale (dBFS(A)). Alternatively, the nominal level may be 13 dB ±2 dB SLR if using the ITU measurement procedure.

Signal-to-Noise Ratio

The accessory must maintain adequate signal-to-noise ratio (SNR) in its uplink output for speech recognition purposes. To maintain high speech recognition accuracy, the absolute SNR should be over 20 dB and all reasonable efforts should be made to maintain high SNR in all typical driving conditions as specified in Annex D of *ITU-T P.1110*. Values above 20 dB are recommended, and the SNR should be at least 25 dB in a stationary vehicle and at slow driving speeds.

Latency

Latency from the reception of sound at the accessory's microphone and the receipt by the CarPlay Communication Plug-in of a buffer of LPCM audio representing that sound must not exceed 60 ms. Likewise, latency from the receipt of a buffer of LPCM audio by the accessory from the CarPlay Communication Plug-in to the time it is emitted by the accessory's speaker must not exceed 35 ms.

Send Frequency Response

The normalized send frequency response for Siri must meet the same requirements as defined for FaceTime. See Main Audio - FaceTime Audio (page 417)

23.2.8.2.5 Main Audio - Alert

The alert audio type is a low latency audio stream.

Sample Rate

Table 23-16 (page 420) details the audio stream requirements. The accessory must support at least one of 44.1 kHz 16-bit or 48 kHz 16-bit audio natively, that is, a sample rate at which the accessory can perform its audio processing without sample rate conversion. See Table 23-28 (page 435) for a list of audio configurations.

Table 23-16 Alert Stream Requirements

Sample Rate	Bit Depth	Channels	Duplexing	Requirement
44.1 kHz	16 bit	mono or stereo	n/a	Required (if 48 kHz 16 bit audio is not supported)

Sample Rate	Bit Depth	Channels	Duplexing	Requirement
48 kHz	16 bit	mono or stereo	n/a	Required (if 44.1 kHz 16 bit audio is not supported)

Latency

Latency from the receipt of a buffer of LPCM audio by the accessory from the CarPlay Communication Plug-in to the time it is emitted by the accessory's speaker must not exceed 35 ms.

23.2.8.2.6 Main Audio - Default

The default audio type is a duplex audio stream.

Sample Rate

Table 23-17 (page 421) details the audio stream requirements. The accessory must accept and provide the required sample rates with equal or higher bit depth.

Table 23-17 Default Audio Stream Requirements

Sample Rate	Bit Depth	Channels	Duplexing	Requirement
16 kHz	16 bit	mono	full-duplex	Required
24 kHz	16 bit	mono	full-duplex	Required
32 kHz	16 bit	mono	full-duplex	Recommended
44.1 kHz	16 bit	mono	full-duplex	Recommended
48 kHz	16 bit	mono	full-duplex	Recommended

Audio Processing

The accessory must not perform echo cancellation or noise processing and must provide full-duplex audio.

Latency

For full duplex audio, the accessory must meet the same latency requirements as defined in Main Audio - Telephony (page 415) and Main Audio - FaceTime Audio (page 417).

23.2.8.2.7 Main High Audio - Entertainment

Main High audio is a high latency audio stream for entertainment audio used only for CarPlay over wireless. It replaces the main audio "entertainment" stream used for CarPlay over USB.

Sample Rate

Table 23-18 (page 422) details the audio stream requirements. The accessory must support at least one of 44.1 kHz 16-bit or 48 kHz 16-bit audio natively, that is, a sample rate at which the accessory can perform its audio processing without sample rate conversion. See Table 23-28 (page 435) for a list of audio configurations.

Table 23-18 Main High Audio Stream Requirements

Sample Rate	Bit Depth	Channels	Duplexing	Requirement
44.1 kHz	16 bit	mono or stereo	n/a	Required (if 48 kHz 16 bit audio is not supported)
48 kHz	16 bit	mono or stereo	n/a	Required (if 44.1 kHz 16 bit audio is not supported)

Latency

Latency from the receipt of a buffer of AAC-LC audio by the accessory from the CarPlay Communication Plug-in to the time it is emitted by the accessory's speaker must not exceed 35 ms.

23.2.8.2.8 Alternate Audio

Sample Rate

Table 23-19 (page 422) details the audio stream requirements. The accessory must support at least one of 44.1 kHz 16-bit or 48 kHz 16-bit audio natively, that is, a sample rate at which the accessory can perform its audio processing without sample rate conversion. See Table 23-28 (page 435) for a list of audio configurations.

Table 23-19 Alternate Audio Stream Requirements

Sample Rate	Bit Depth	Channels	Duplexing	Requirement
44.1 kHz	16 bit	mono or stereo	n/a	Required (if 48 kHz 16 bit audio is not supported)
48 kHz	16 bit	mono or stereo	n/a	Required (if 44.1 kHz 16 bit audio is not supported)

Latency

Latency from the receipt of a buffer of LPCM audio by the accessory from the CarPlay Communication Plug-in to the time it is emitted by the accessory's speaker must not exceed 35 ms.

23.2.8.2.9 Volume Management

The accessory's volume controls (physical or on-screen) must control the volume of audio from the Apple device.

If the accessory supports maintaining separate volumes for different types of audio, it is recommended that it does so using the following volume categories, with corresponding mappings to main audio stream audioTypes (see Table 23-40 (page 443)), and alternate audio:

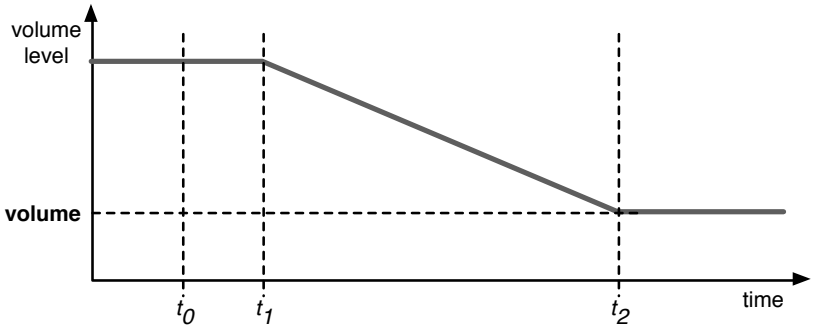
- Entertainment - main audio "media"
- Ringtone - main audio "alert"
- Phone Calls - main audio "telephony"
- Voice Prompts and Notifications - main audio "speech recognition" and alternate audio
- Other - main audio "default"

For example, were the user to change the volume while listening to music from the device, the accessory would remember that volume for the Entertainment volume category, and subsequently restore that volume the next time the main audio audioType was configured for "default" or "media".

23.2.8.2.10 Ducking

The accessory must support ducking, the process of ramping the main audio volume down so the user can more clearly hear an announcement or alert from the alternate audio channel. This ramping must be free of any zippering or other audible artifacts. For use cases where ducking is required, the device will send a "duckAudio" command to the accessory. For example, if the user is listening to music and they are getting close to their destination, the music may be ducked (attenuated) so that another stream can say "Arriving at your destination." Ducking fades the main audio volume down to a lower level so it is not an abrupt change. When the announcement is done, the original stream is unducked by fading back to the original level.

Figure 23-7 Event timeline for audio ducking/unducking



t_0: duckAudio command received by accessory
t_1: accessory audio engine ready to begin volume ramp
t_2: volume ramp ends; t_2 - t_0 = **durationMs**

The duck/unduck ramp (t1 - t0) must start within a very short time after the message has been received. Typical and recommended times are 10-20 ms; response within 75 ms is required.

23.3 CarPlay Communication Protocol

The CarPlay communication protocol provides a mechanism for an accessory and an Apple device to:

- Discover and connect to one another on a network
- Setup and control audio and video (UI) streaming
- Arbitrate ownership and use of the accessory's shared resources
- Communicate the Apple device's app state to avoid conflicts that would lead to undesirable user experiences.
- Exchange user input events

Implementation of this protocol must incorporate the reference implementation provided in the CarPlay Communication Plug-in.

The CarPlay communication protocol is flexible in its parsing of parameter values. CarPlay message parameters that are numbers, enum values, or booleans may be provided as strings that parse as those data types.

23.3.1 Discovery

This section contains requirements for Apple device discovery by accessories as well as accessory discovery by Apple devices.

23.3.1.1 Apple Device Discovery by Accessories

Apple devices supporting CarPlay will advertise a Bonjour service to the accessory. The advertised services is of a server type `_carplay-ctrl._tcp`. The service provides an HTTP-based protocol to initiate a CarPlay session with an Apple device.

Accessories must use this service to discover CarPlay enabled devices and to initiate the CarPlay session to a specific device. The accessory may use the service to display a list of CarPlay enabled devices to the user.

> **Note:** Apple devices which do not support the CarPlay Control service (iOS 8.2 or older), will initiate the CarPlay session automatically.

The CarPlay Communication Plug-in implements the Bonjour discovery and provides a command protocol to the accessory. See CarPlay Control (page 425).

The name of the Bonjour service is the user-visible name of the Apple device (e.g. "Tim's iPhone"). The name may contain any Unicode character and is encoded using UTF-8. It has a maximum length of 63 bytes (which may be fewer than 63 characters as a single Unicode character may require multiple bytes). Additional metadata needed at discovery-time is advertised via a TXT record as defined in Table 23-20 (page 425).

Table 23-20 CarPlay Control Bonjour Service Keys

Key	Description
id	Bluetooth MAC address for the Apple device; e.g., "00:11:22:33:44:55".
srcvers	Apple CarPlay source version assigned by Apple, in the form "x.y.z".

23.3.1.1.1 CarPlay Control

When the accessory wants to send a CarPlay Control command, it must do the following:

1. Look up the DNS name of the Apple device with Bonjour by resolving the _carplay-ctrl._tcp service.
2. Resolve the DNS name to the IP address(es) of the CarPlay Control Apple device. If the accessory platform's `getaddrinfo` is Bonjour aware then it may be used. Otherwise, Bonjour's asynchronous DNSServiceGetAddrInfo API should be used.
3. Connect to the CarPlay Control Apple device's IP address(es). There may be multiple IP addresses associated with the DNS name. The accessory should try each IP address until it makes a successful TCP connection (or runs out of IP addresses and fails). IPv6 addresses are generally preferred because in some environments, the Apple device and the accessory may be on different IPv4 subnets, even if they are on the same physical

link. Prioritizing link-local IPv6 addresses over IPv4 addresses often means a faster and more reliable connection. Link-local addresses should not be used as they will not be able to wake an Apple device that is in a sleep state.

4. Send the command using an HTTP request.

CarPlay Control requests look like normal HTTP requests. The request line of the request uses the following format to specify the command:

```
GET /ctrl-int/1/<command> HTTP/1.1
```

The command must be one of the following:

Table 23-21 CarPlay Control Commands

Command	Description
connect	Requests that the Apple device begin a CarPlay session with the accessory issuing the request.

All command requests must include the `AirPlay-Receiver-Device-ID` header field. This header field specifies the accessory that the Apple device should connect to. The value of this field is the primary MAC address of the accessory as a 48-bit integer.

For example, if an accessory wanted a Apple device to initiate a CarPlay session to it then it would send the following HTTP request to the Apple device:

```
GET /ctrl-int/1/connect HTTP/1.1
Host: bb15.local.
AirPlay-Receiver-Device-ID: 18838586676582
```

23.3.1.2 Accessory Discovery by Apple Devices

CarPlay accessories must advertise a Bonjour service to the Apple device. The advertised service must be of a server type `_airplay._tcp`. Apple devices browse for these accessories and offer them to the user if they are found and determined to be compatible.

Additional metadata needed at discovery-time is advertised via a TXT record containing fields for feature detection, model information, versions, etc. as defined in Table 23-22 (page 427). The CarPlay Communication Plug-in will interface to the accessory's Bonjour implementation to configure and broadcast these values.

Table 23-22 CarPlay Bonjour Service Keys

Key	Required?	Description
deviceid	Y	Globally unique ID for the accessory; e.g., the primary MAC address, such as "00:11:22:33:44:55".
features	Y	Internally defined for Apple CarPlay. Value must not be modified.
model	Y	Model name of the accessory; must be globally unique and uniquely identify the product. This should be the model name of the vehicle for OEM integrations or an aftermarket unit's model name.
protovers	N	Protocol version string <major>.<minor> (e.g., "1.0"). Missing key defaults to "1.0".
srcvers	Y	Apple CarPlay source version assigned by Apple, in the form "x.y.z".
flags	Y	Status of an operation. Value must not be modified. See Table 23-23 (page 427).

Table 23-23 Status flags bitfield enum values

Value	Bit	Description
0x01	0	Problem has been detected.
0x02	1	Device is not configured.
0x04	2	Audio cable is attached.

The accessory must enable the Bonjour DirectLink optimization. This will optimize the Bonjour discovery process and speed up the connection establishment time over the NCM interface.

23.3.2 Setup and Control

The Communication Plug-in in the CarPlay Client establishes a control channel for the primary tasks of:

- Configuring the UI stream to one of the target screen resolutions listed in Media Types and Formats (page 411).
- Scheduling the video signal to a synchronized clock between both accessory and Apple device (a microsecond resolution clock or better is required on the accessory).
- Configuring and managing the main and alternate audio streams, including callbacks to schedule audio delivery to the accessory and notifications from the Apple device whenever sample rate or format changes are required.

- Synchronizing shared resources such as the accessory's screen and speakers, plus determining which media source currently has audio and/or video focus on those resources.

HTTP/RTSP is used to set up, control, and monitor a CarPlay session. When an Apple device discovers a compatible accessory that is either already known by the Apple device or is user selected, the Apple device initiates a connection to the accessory and starts a CarPlay session. The following summarizes a typical session lifetime:

1. The Apple device establishes a TCP control connection to the accessory.
   - This connects to the TCP port advertised via Bonjour.
2. The Apple device authenticates the accessory and sets up a secure channel.
   - This process requires the accessory to perform the message exchange described in MFI-SAP (page 429).
   - After this step, all subsequent control requests and responses are encrypted.
3. The Apple device sends an info request to provide its info and get the accessory's info.
   - The request message is a plist providing the Apple device's model, version, etc.
   - The response message is a plist returning the accessory's audio formats, display info, model, current modes, etc.
   - See Info Message (page 430).
4. The Apple device sends a setup request to set up streams for control and, if needed, audio, screen, etc.
   - This describes each stream to set up, including any TCP/UDP port numbers, configuration options, etc.
   - The accessory responds with details about the streams it set up, such as listening TCP/UDP port numbers.
   - For input or output TCP streams, the accessory will start listening for TCP connections.
   - For input UDP streams, the Apple device will start listening for UDP packets.
   - For output UDP streams, the accessory will start listening for UDP packets.
   - See Setup Message (page 440).
5. The Apple device makes a TCP connection to the accessory for each TCP-based stream.
   - For both input and output streams, the Apple device initiates the connection.
   - For output streams, the Apple device sends data to the accessory (e.g., music to play on the accessory).
   - For input streams, the accessory will send data to the Apple device (e.g., HID reports). See Table 23-40 (page 443) and Table 23-43 (page 444).
6. The Apple device sends a record request to start a session on the accessory.

- The accessory performs time sync negotiation with the Apple device. See Synchronization (page 447).
- The accessory starts corresponding streams, creating threads to process audio/video data, timers, etc.

7. The Apple device sends setup requests to set up streams for audio and/or screen, and makes TCP connections as needed.
8. The Apple device and accessory exchange data.
   - The Apple device sends audio and video output data to the accessory for playback.
   - The accessory sends audio input data (if any) to the Apple device for input.
9. The Apple device sends a TEARDOWN requests to end one or more streams on the accessory.
   - The accessory stops processing for specified streams and releases corresponding resources.
10. The Apple device sends a TEARDOWN request to end the session on the accessory.
    - The accessory stops all audio and video processing and releases all session-specific resources.
11. The Apple device closes all TCP connections to the accessory.
    - This resets the Apple device back to the point before it started the session.

23.3.2.1 MFI-SAP

Authentication over the CarPlay interface requires use of the MFi Authentication Coprocessor for authenticating an encrypted channel. The protocol uses Curve25519 Elliptic-Curve Diffie-Hellman technology for key exchange, as described in cr.yp.to/ecdh.html. It also uses RSA for signing and verifying and AES-128 in counter mode for encryption.

Authentication starts when the Apple device generates a new, random Curve25519 key pair and sends its Curve25519 public key X to the accessory. The accessory responds by sending its public key Y to the Apple device, followed by its MFi Certificate and an encryption of Signature(Y,X).

Messages are exchanged using RTSP/HTTP POST requests and responses to the `/auth-setup` URL. The content of the message is delivered via the body of the RTSP/HTTP message.

When the accessory receives the authentication request, it must:

1. Generate a new, random Curve25519 key pair.
2. Generate the shared secret using its Curve25519 private key and the streamer's Curve25519 public key.
3. Sign the two public keys with its RSA private key and encrypt the result with the AES master key derived from the Curve25519 shared secret.
4. Send its Curve25519 public key, its MFi certificate, and its AES-encrypted signature to the Apple device.

When the Apple device receives the response, it verifies that the accessory's MFi certificate is valid, decrypts the accessory response and verifies that the signature was signed by that certificate. If the signature verifies, the accessory is considered authenticated. The AES key derived from the Curve25519 shared secret is then used to encrypt all future content.

All Curve25519 keys must be generated from cryptographically strong random numbers. New Curve25519 keys must be generated for each authentication attempt; reuse of keys by the accessory is not permitted.

Each media stream must use a unique key and IV for encrypting content to avoid reusing the same key/IV for different pieces of data. The following are used to derive stream keys and IVs:

- ekey = Bytes 0-15 of SHA-512("AirPlayStreamKey" + streamConnectionID + session key (master))
- eiv = Bytes 0-15 of SHA-512("AirPlayStreamIV" + streamConnectionID + session key (master))

The steam connection ID is contained in Table 23-43 (page 444), and the ekey, eiv, and encryption type are contained in Table 23-36 (page 441). Table 23-24 (page 430) details the values used to specify the encryption type.

Table 23-24 Encryption type enum values

Value	Description
0x00	Invalid.
0x01	Unencrypted.
0x10	MFi-SAP-encrypted AES key.

23.3.2.2 Info Message

The info message provides Apple device information to the accessory and returns accessory information to the Apple device. It is an HTTP GET to the `/info` URL (see Table 23-50 (page 446)) with binary plist request and response payloads. The Apple device may include any information it wants to provide to the accessory in the request plist. The request (see Table 23-25 (page 431)) may contain the "qualifier" key to limit the properties being requested from the accessory; for example, the Apple device may only want to get the accessory's model. The response (see Table 23-26 (page 431)) contains the properties requested by the Apple device.

Table 23-25 Info Message request keys

Key	Type	Required?	Description
qualifier	array	N	Array of property strings from Table 23-26 (page 431) to request from the accessory. If this key is missing, the accessory must return all of its properties. If this key is empty, the accessory must not return any properties.

Table 23-26 Info Message response keys

Key	Type	Required?	Description
audioFormats	array	Y	Array of dictionaries for audio formats supported by the accessory. Accessories must provide the supported sample rates for each audioType for all audio streams. In order to maintain compatibilty with versions of iOS that do not support CarPlay over wireless, the first two entries in this array must be: 1) type=100, audioType="compatibility", and must only contain PCM formats 2) type=101, audioType="compatibility", and must only contain PCM formats. These first two entries will be used by earlier versions of iOS that do not directly support the "audioType" key. Following the previous two entries, there must be entries for all valid combinations of type and audioType keys. See Table 23-27 (page 435).
audioLatencies	array	Y	Array of dictionaries specifying the fixed audio latencies within the accessory. See Table 23-29 (page 437). Accessory must specify default latencies for all audio streams, audio types and audio formats (some entries can be omitted based on the optional specifiers). A latter entry in the array overrides an earlier one if they both apply for a given stream type/audio type/format (last one wins).

Key	Type	Required?	Description
bluetoothIDs	array	N (*)	Array of MAC address strings for the Bluetooth modules in the accessory. MAC addresses must be provided in colon separated format, for example "00:11:22:33:44:55". See disableBluetooth (page 478). (*) Required when accessory supports Bluetooth.
deviceID	string	Y	Globally unique device ID: e.g., "00:11:22:33:44:55". See Table 23-22 (page 427).
displays	array	Y	An array of dictionaries for display information supported by the accessory. See Table 23-30 (page 437).
extendedFeatures	array	N (*)	Array of strings indicating support for additional features. See Table 23-33 (page 439). (*) Required when accessory supports CarPlay over wireless.
features	bitfield	Y	Internally defined for Apple CarPlay. Value must not be modified. See Table 23-22 (page 427).
firmwareRevision	string	N	Firmware revision of the accessory, e.g. "FirmwareRevision0.1".
hardwareRevision	string	N	Hardware revision of the accessory, e.g. "HardwareRevision0.1".
hidDevices	array	Y	An array of dictionaries for HID device information. See User Input (page 456).
hidLanguages	array	N	List of BCP-47 language code strings for languages supported by the accessory's character recognizer.
keepAliveLowPower	boolean	Y	Indicates if the accessory supports session idle UDP keepalive.

Key	Type	Required?	Description
keepAliveSendStatsAsBody	boolean	Y	Indicates whether the accessory supports statistics as part of the keep alive message body. This is used for debugging.
limitedUIElements	array	N (*)	Array of UI elements affected by limited UI mode. See Table 23-34 (page 439). (*) Required when limitedUI.
limitedUI	boolean	N	Indicates whether or not certain UI elements are limited. See setLimitedUI (page 484).
macAddress	string	N	MAC address of the accessory's local network interface. See iAP2 / NCM Interface Configuration (page 389). This address must be different from the one which is provided to the Apple device via the CDC Ethernet Networking functional descriptor.
manufacturer	string	Y	Name of the accessory manufacturer (user-visible).
model	string	Y	The model name of the accessory, which must be globally unique and must uniquely identify the product. This should be the model name of the vehicle for OEM integrations or an aftermarket unit's model name. See Table 23-22 (page 427).
modes	group	Y	The modes describing the current state of the accessory, including appState, mainScreen and mainAudio as applicable. See the technical notes in the CarPlay DevKit for more details. The modes dictionary uses the same keys as the "changeModes" command, see Table 23-67 (page 479).
name	string	Y	This must be set to "CarPlay".

Key	Type	Required?	Description
nightMode	boolean	N	Indicates if it is dark outside and appearance should change to suit night time use. If the accessory does not support night mode, then it should not respond to this property at all (return an unsupported error). The Apple device can then use other metrics (such as time of day) to decide on its own when to enter night mode. See setNightMode (page 484).
oemIcons	array	N (*)	An array of dictionaries for icons representing the accessory manufacturer's logo. Icons should be provided for the following pixel sizes: 120x120, 180x180, and 256x256. The Apple device will use the image sized most appropriately for the usage and accessory screen size. See Table 23-35 (page 440). (*) Required when oemIconVisible is True.
oemIconLabel	string	N (*)	Label shown underneath the oemIcon. (*) Required when oemIconVisible is True.
oemIconVisible	boolean	N	Whether or not the oemIcon is visible on the home screen.
OSInfo	string	N	Operating system information including name, version, and architecture (e.g. "Darwin 13.0.0 x86_64").
protocolVersion	string	Y	Protocol version string major.minor ; e.g., "1.0". The default value is "1.0". See Table 23-22 (page 427).
rightHandDrive	boolean	Y	True if the vehicle is right hand drive.
sourceVersion	string	Y	CarPlay source version assigned by Apple, in the form x.y.z . See Table 23-22 (page 427).
statusFlags	bitfield	Y	Status of an operation. Value must not be modified. See Table 23-22 (page 427).

Table 23-27 Audio format keys

Key	Type	Required?	Description
type	enum	Y	The stream type to which the input and/or output formats apply. One of Table 23-45 (page 444).
audioInputFormats	bitfield	Y (*)	The audio input formats supported by the given type. See Audio (page 413) for requirements. (*) Not required if the stream type does not support input. Combination of Table 23-28 (page 435).
audioOutputFormats	bitfield	Y	The audio output formats supported by the given type. See Audio (page 413) for requirements. Combination of Table 23-28 (page 435).
audioType	string	N	The audio type to which the format(s) apply. See Table 23-46 (page 445).

The audio formats for wired and wireless transports must be combined in a single entry. PCM formats must be used for wired and compressed formats must be used for wireless. Multiple compression type (e.g. AAC-ELD and OPUS) must not be offered simultaneously.

The audio input formats supported by the accessory must be a subset of the audio output formats supported. When using full-duplex audio, the Apple device will use the same format for both directions and will choose from the supported input formats.

Table 23-28 Audio formats bitfield enum values

Value	Bit	Description
0x00000004	2	PCM, 8000 Hz, 16-Bit, Mono
0x00000008	3	PCM, 8000 Hz, 16-Bit, Stereo
0x00000010	4	PCM, 16000 Hz, 16-Bit, Mono
0x00000020	5	PCM, 16000 Hz, 16-Bit, Stereo
0x00000040	6	PCM, 24000 Hz, 16-Bit, Mono
0x00000080	7	PCM, 24000 Hz, 16-Bit, Stereo
0x00000100	8	PCM, 32000 Hz, 16-Bit, Mono
0x00000200	9	PCM, 32000 Hz, 16-Bit, Stereo

Value	Bit	Description
0x00000400	10	PCM, 44100 Hz, 16-Bit, Mono
0x00000800	11	PCM, 44100 Hz, 16-Bit, Stereo
0x00001000	12	Reserved
0x00002000	13	Reserved
0x00004000	14	PCM, 48000 Hz, 16-Bit, Mono
0x00008000	15	PCM, 48000 Hz, 16-Bit, Stereo
0x00010000	16	Reserved
0x00020000	17	Reserved
0x00040000	18	Reserved
0x00080000	19	Reserved
0x00100000	20	Reserved
0x00200000	21	Reserved
0x00400000	22	AAC-LC, 44100 Hz, Stereo
0x00800000	23	AAC-LC, 48000 Hz, Stereo
0x01000000	24	AAC-ELD, 44100 Hz, Stereo
0x02000000	25	AAC-ELD, 48000 Hz, Stereo
0x04000000	26	AAC-ELD, 16000 Hz, Mono
0x08000000	27	AAC-ELD, 24000 Hz, Mono
0x10000000	28	OPUS, 16000 Hz, Mono
0x20000000	29	OPUS, 24000 Hz, Mono
0x40000000	30	OPUS, 48000 Hz, Mono
0x80000000	31	Reserved

The PCM formats must be only used for CarPlay over USB. For CarPlay over wireless, the AAC-LC compression must only be used for the high latency "main high audio - entertainment" stream, whereas AAC-ELD or OPUS compression must be used for the low latency "main audio" streams (except for "entertainment" audio) and alternate audio. See Audio (page 413).

Table 23-29 Audio latencies keys

Key	Type	Required?	Description
type	enum	Y	The stream type to which the input and/or output formats apply. One of Table 23-45 (page 444).
audioType	string	Y	Type of audio content (e.g. telephony, media, etc.). See Table 23-46 (page 445).
sr	number	N	Number of samples per second (e.g. 44100).
ss	number	N	Bit size of each audio sample (e.g. "16").
ch	number	N	Number of audio channels (e.g. 2 for stereo).
inputLatencyMicros	number	Y (*)	Input latency in microseconds. (*) Only required for duplex audio streams.
outputLatencyMicros	number	Y	Output latency in microseconds.

Table 23-30 Display keys

Key	Type	Required?	Description
edid	data	N	Raw EDID of the display, if available.
features	bitfield	Y	Features of the display as a bitmask. Combination of Table 23-31 (page 438).
maxFPS	number	N	Max frames per second the display supports.
heightPixels	number	Y	Height of the display in pixels. Must be non-zero.
widthPixels	number	Y	Width of the display in pixels. Must be non-zero.
heightPhysical	number	Y	Height of the display in millimeters. Must be non-zero.
widthPhysical	number	Y	Width of the display in millimeters. Must be non-zero.

Key	Type	Required?	Description
uuid	string	Y	UUID of the display.
primaryInputDevice	enum	Y	Primary input device to be used for navigating the user interface. One of Table 23-32 (page 438).

Table 23-31 Display features bitfield enum values

Value	Bit	Description
0x02	1	Supports interacting via knobs.
0x04	2	Supports interacting via low-fidelity touch.
0x08	3	Supports interacting via high-fidelity touch.
0x10	4	Supports interacting via touchpad.

The display features value is a bitmask, so combinations are possible. It is required to set the appropriate bit for all input devices supported by the accessory.

To claim support for high-fidelity touch, the time between a user touch input to the time a frame updates on the display must be less than 140 ms.

Table 23-32 Primary input device enum values

Value	Description
0x01	Accessory uses touchscreen as primary input.
0x02	Accessory uses touchpad as primary input.
0x03	Accessory uses knob as primary input.

Note that the choice of the primary input device, along with the display features supported by the accessory, will directly impact the look and feel of the user interface presented by the Apple device on the accessory's display:

- An accessory that reports supporting low-fidelity touch (display feature bit 2) with touchscreen as the primary input device (value 0x01) will be presented with a limited touch interface with other input methods as secondary inputs. For instance, the user interface for a map will be augmented with soft arrow-keys as overlays to enable panning.

- An accessory that reports supporting high-fidelity touch (display feature bit 3) with touch as the primary input device (value 0x01) will be presented with a touch interface that allows for scrolling and panning gestures with other input methods as secondary inputs.
- An accessory that reports touchpad (display feature bit 4) as primary input device (value 0x02) will be presented with a user interface that may be interacted with primarily via the use of touchpad, select, and back buttons with other input methods as secondary inputs.
- An accessory that reports knob (display feature bit 1) as primary input device (value 0x03) will be presented with a user interface that may be interacted with primarily via the use of knob, select, and back buttons with other input methods as secondary inputs.

Table 23-33 Extended features string values

Value	Description
"enhancedRequestCarUI"	Indicates accessory support for the "url" request key of the requestUI command. See requestUI (page 483).
"vocoderInfo"	Provides the native sample rate of the telephony output stream. As an optimization, the Apple device may prefer to encode the stream at a sample rate other than that used by the cellular network. The accessory may use this parameter to optimize telephony audio playback to the appropriate sample rate. Required when accessory supports CarPlay over wireless.

Table 23-34 Limited UI elements string values

Value	Description
"softKeyboard"	Touch keyboard that appears on screen.
"softPhoneKeypad"	Touch phone keypad that appears on screen.
"nonMusicLists"	Lists of non-music items.
"musicLists"	Lists of music items.
"japanMaps"	Minor roads in Japan.

Table 23-35 Icon keys

Key	Type	Required?	Description
imageData	data	Y	PNG data for an icon. Icon can be either a mask or a full-bleed image. Mask icons should be solid white with a transparent, alpha channel background. Full-bleed icons should be 8 bits per channel and must not include an alpha channel. Avoid including icon outlines, shine, or other effects. The Apple device will clip the icon to the appropriate shape and provide any additional styling required to match the design principles of the current version of iOS. The icon should follow the design principles of the current version of iOS. For example, see the Mail envelope icon.
heightPixels	number	Y	Height in pixels of the image.
widthPixels	number	Y	Height in pixels of the image.
prerendered	boolean	Y	Indicates if the icon is full-bleed. False if the icon is a mask icon.

The `/info` URL may also be used via an HTTP GET to read accessory information. The URL may contain a query string to limit the properties being requested (for example, `GET /info?deviceID&model` to get the accessory's device ID and model properties).

23.3.2.3 Setup Message

The setup message is sent by the device to configure and set up all streams between the accessory and the device. The CarPlay streams are logical flows of data, such as a control stream for sending commands and getting responses, a screen stream for sending H.264 frames, an audio stream for sending audio, etc. The setup request message includes descriptions of all the streams the Apple device wants to set up. The response includes details about the streams that were set up, i.e. control only, main audio and screen streams, alternate audio only, etc. Both the request and the response payloads are binary plists.

The initial setup message after the device connects to the accessory contains information to set up the control stream. If needed, it may also setup the audio and screen streams.

Table 23-36 (page 441) lists the specifics of the initial setup request keys sent by the Apple device to the accessory.

Table 23-36 Initial Setup request keys

Key	Type	Required?	Description
deviceID	string	Y	Globally unique device ID (e.g. "11:22:33:44:55").
eiv	data	Y	AIS Encryption Initialization Vector for the accessory. This is sent in the clear. See MFI-SAP (page 429)
ekey	data	Y	AIS Encryption Key for the accessory. Encrypted with master key from encryption setup. See MFI-SAP (page 429)
et	enum	Y	Encryption type. One of Table 23-24 (page 430).
macAddress	string	Y	MAC address of the iOS network interface used for the connection.
model	string	Y	Model name of the Apple device (e.g. "Device1,1").
name	string	Y	User-customizable name of the Apple device.
osBuildVersion	string	Y	Operating system build version string (e.g. "11A200").
sessionUUID	string	Y	UUID of the session.
sourceVersion	string	Y	Apple device side CarPlay source version string (e.g. "101.7").
streams	array	Y (*)	Array of stream descriptions. See Table 23-40 (page 443) and Table 23-43 (page 444). (*) The initial setup message may not include "streams" if it only sets up the control stream.
timingPort	number	Y	Port the Apple device is listening on for time sync requests.

Table 23-37 (page 441) lists the specifics of the initial setup response keys sent by the accessory to the Apple device.

Table 23-37 Initial Setup response keys

Key	Type	Required?	Description
eventPort	number	Y	TCP/UDP port number for events.
keepAlivePort	number	Y	UDP port number for session idle UDP keepalive.

Key	Type	Required?	Description
streams	array	Y (*)	Array of stream descriptions. See Table 23-40 (page 443) and Table 23-43 (page 444). (*) The initial setup message may not include "streams" if it only sets up the control stream.
timingPort	number	Y	Port the Apple device is listening on for time sync requests.

Once a session is established, the setup message is sent to set up and configure the audio and screen streams, as they are required by the device.

Table 23-38 (page 442) lists the specifics of the setup request keys sent by the Apple device to the accessory after a session has been started.

Table 23-38 Setup request keys

Key	Type	Required?	Description
streams	array	Y	Array of stream descriptions. See Table 23-40 (page 443) and Table 23-43 (page 444).

Table 23-39 (page 442) lists the specifics of the setup response keys sent by the accessory to the Apple device after a session has been started.

Table 23-39 Setup response keys

Key	Type	Required?	Description
streams	array	Y	Array of stream descriptions. See Table 23-40 (page 443) and Table 23-43 (page 444).

23.3.2.3.1 Streams

Each stream is set up using a stream descriptor with a type, parameters, and behaviors associated with it, as listed in Table 23-40 (page 443) for main/alternate audio stream and Table 23-43 (page 444) for screen stream.

The timestamp and sampleTime response parameters are used for media clock synchronization of the main and alternate audio streams as described in Synchronization (page 447).

Table 23-40 Main/alternate audio stream descriptors request keys

Key	Type	Required?	Description
audioFormat	bitfield	Y	Format of the audio data (e.g. 44100 Hz, Stereo PCM). For main audio applies to both input and output. Combination of Table 23-28 (page 435).
audioLatencyMs	number	Y	Desired milliseconds of audio latency. This value is used to set the size of the network jitter buffer and is therefore the largest possible size that can be accomodated by a read or write of audio data. The IO block size used by the accessory to read and write audio data must be no larger than this size and it is strongly recommended that it be smaller in order to avoid under or overflowing the jitter buffer.
audioLoopback	boolean	N	True if audio output must be looped back and sent as audio input. Debug use only.
audioType	string	Y (*)	Type of audio content (e.g. telephony, media, etc.). See Table 23-46 (page 445). (*) Main audio only.
dataPort	number	Y (*)	UDP port the Apple device is listening on for data. (*) Only required when audio input is enabled (main audio only).
input	boolean	N	True if input must be enabled for the stream (main audio only).
type	enum	Y	Type of stream. One of Table 23-45 (page 444).
vocoderInfo	group	Y	Telephony vocoder information. Contains dictionary of Table 23-41 (page 443).

Table 23-41 Vocoder Information keys

Key	Type	Required?	Description
sampleRate	number	Y	The native sample rate of the telephony output stream.

Table 23-42 Main/alternate audio stream descriptors response keys

Key	Type	Required?	Description
dataPort	number	Y	UDP port the accessory is listening on for data.

Key	Type	Required?	Description
type	enum	Y	Type of stream. One of Table 23-45 (page 444).
sampleTime	number	N	Media timestamp at the same moment as "timestamp".
timestamp	number	N	Wall clock timestamp at the same moment as "sampleTime" using synchronized wall time.

Table 23-43 Screen stream descriptors request keys

Key	Type	Required?	Description
latencyMs	number	Y	Desired milliseconds of screen latency.
type	enum	Y	Type of stream. One of Table 23-45 (page 444).
streamConnectionID	number	Y	Unique stream connection ID. Used for deriving encryption key and IV. See MFI-SAP (page 429)

Table 23-44 Screen stream descriptors response keys

Key	Type	Required?	Description
dataPort	number	Y	TCP or UDP port the accessory is listening on for data.
type	enum	Y	Type of stream. One of Table 23-45 (page 444).

Table 23-45 Stream ID enum values

Name	Value	Protocol	Description
Invalid	0	n/a	Reserved for an invalid stream ID.
Main Audio	100	UDP	Low-latency audio input/output. Value is also the RTP payload type.
Alt Audio	101	UDP	Low-latency UI sounds, alerts, etc., output. Value is also the RTP payload type.
Main High Audio	102	UDP	High-latency audio output. Value is also the RTP payload type.
Screen	110	TCP/UDP	H.264 screen output. Value is also the RTP payload type.

Table 23-46 Audio type string values

Value	Stream ID	Description
"default"	Main Audio, Alt Audio	Unspecified or unknown audio type.
"alert"	Main Audio	Ringtones, alarms, and other high-priority sounds.
"media"	Main Audio, Main High Audio	Entertainment (e.g.: music playback).
"telephony"	Main Audio	Telephony.
"speechRecognition"	Main Audio	Speech recognition.
"compatibility"	Main Audio, Alt Audio	Special value used only in the Info Message response for specifying audio formats compatible with older versions of iOS. Not a valid audio type for stream descriptors.

Table 23-47 Channels

Name	Transport	Port	QoS	Direction	Description
RTSP Controller control	TCP	5000	BE	device to accessory	iOS control channel
RTSP Accessory control	TCP	dynamic	BE	accessory to device	Accessory control channel
RTP Main Audio Output	UDP	dynamic	VO	device to accessory	Audio data
RTP Alternate Audio Output	UDP	dynamic	VO	device to accessory	Audio data
RTP Main Audio Input	UDP	dynamic	VO	accessory to device	Audio data
Time Sync Client	UDP	dynamic	CTL	accessory to device	Audio time sync requests
Time Sync Server	UDP	dynamic	CTL	device to accessory	Audio time sync responses
RTP Screen	TCP	dynamic	VI	device to accessory	User interface (H.264 frames)

23.3.2.4 Feedback Message

The feedback message exchanges timing information and statistics to and from an accessory. The Apple device does an HTTP POST to the `/feedback` URL and the accessory responds with its feedback info. The request and response payloads are binary plists. These plists may be omitted or empty if they don't have feedback to report in that direction.

This message is used for media clock synchronization of the main and alternate audio streams as described in Synchronization (page 447).

Table 23-48 Feedback response keys

Key	Type	Required?	Description
streams	array	N	Streams reporting feedback. Contains dictionaries of Table 23-49 (page 446).

Table 23-49 Feedback stream keys

Key	Type	Required?	Description
sr	number	N (*)	Estimated consumption rate in samples per second of the stream. This is calculated based on sampleTime and timestamp. (*) Required when type is an audio stream.
type	enum	Y	Type of stream. One of Table 23-45 (page 444).
sampleTime	number	N	Media timestamp at the same moment as "timestamp".
timestamp	number	N	Wall clock timestamp at the same moment as "sampleTime" using synchronized wall time.

23.3.2.5 URLs

Table 23-50 (page 446) lists the URLs used with CarPlay. All URLs are accessed via HTTP/RTSP.

Table 23-50 URLs for CarPlay

URL	Method	Description
n/a	OPTIONS	Standard HTTP options message to return the supported methods, etc.
n/a	RECORD	RTSP message to start playback.

URL	Method	Description
n/a	SETUP	RTSP message to set up a session and streams.
n/a	TEARDOWN	RTSP message to tear down a session.
`/auth-setup`	POST	Performs MFi-SAP authentication; see MFI-SAP (page 429). Request and response are opaque security data.
`/command`	POST	For performing command requests and their responses; see Commands (page 477). Requests and responses are binary plists.
`/feedback`	POST	Exchanges information with the accessory; see Feedback Message (page 446). Requests and responses are binary plists.
`/info`	GET	Exchanges information with the accessory; see Info Message (page 430). Requests and responses are binary plists.
`/perf`	POST	Performs network performance tests. Requests and responses are binary plists.

23.3.2.6 Synchronization

The clocks between the Apple device and the accessory must be synchronized to present media data at the correct time. The synchronization process is managed by the Communication Plug-in in the CarPlay Client and includes:

- Synchronizing wall clocks so both entities have an accurate notion of absolute presentation times (for example, when the user should hear an audio sample). Note that the synchronized clocks are maintained internally and do not affect the system time of either the Apple device or the accessory.
- Mapping media clocks to wall clocks so that a media timestamp from one device can be mapped to a media timestamp on another device. This allows the media production or consumption rate of one device's media to be clocked off the other device.

The accessory must drive all audio streams, regardless of type (input or output) from the same media clock that is used for synchronization with the Apple device.

23.3.2.6.1 Wall Clock Synchronization

Synchronizing wall clocks is done by exchanging Network Time Protocol (NTP) packets. The Apple device acts as the NTP server, providing the reference clock, and the accessory acts as the NTP client, syncing its clock to the reference. There are two parts to this process: rate and offset synchronization. Rate synchronization tracks how fast the reference clock is running relative to the local clock. This is done by measuring the difference between timestamps over an increasingly large interval to minimize measurement noise (such as network

jitter). The interval is capped to allow it to adapt to real changes in the reference clock rate (for example, from temperature changes affecting the rate). Offset synchronization tracks the absolute difference from the reference clock. This is measured with each packet exchange and smoothed to minimize measurement noise.

23.3.2.6.2 Media Clock Synchronization

Synchronizing media clocks is done by periodically exchanging mapping relationships between media sample time and wall clock time. When the accessory is providing the clock, it must sample its media clock and synchronized wall clock at the same moment. These samples are used to calculate media clock rate. The synchronized wall clock is used so the calculated rate is based on time relative to the Apple device. This allows the Apple device to adapt its media production rate to match the accessory's consumption rate and avoid the need for sample rate conversion. The media clock and wall clock tuple also allows the Apple device to determine the absolute media clock offset between devices. Media clock timing relationships are exchanged via the Feedback Message (page 446).

The parameters exchanged in the Feedback message are calculated by the Communication Plug-in on behalf of the accessory. To do so, the accessory must provide local and media clocks using a high resolution tick counter to measure time intervals (such as a processor cycle counter). These counters must be stable (less than 50 PPM of variance) and precise (10 MHz or higher) with minimal and consistent overhead (less than 1 microsecond to sample each clock).

23.3.2.7 Keepalive

CarPlay uses a combination of methods to monitor the session status. When either screen stream or audio stream is active, the controller uses feedback messages sent over the control channel as "keepalive" messages. When the screen stream is active, the controller also sends screen-specific keepalive messages via the screen data connection. If the accessory supports low power keepalive mode (as indicated by "keepAliveLowPower" feature flag), the controller switches to UDP-only keepalive messages when there are no active media streams to save power.

23.3.3 Resource Management

23.3.3.1 Resources

In its typical embodiment in an automobile, the CarPlay architecture includes three resources that must be managed:

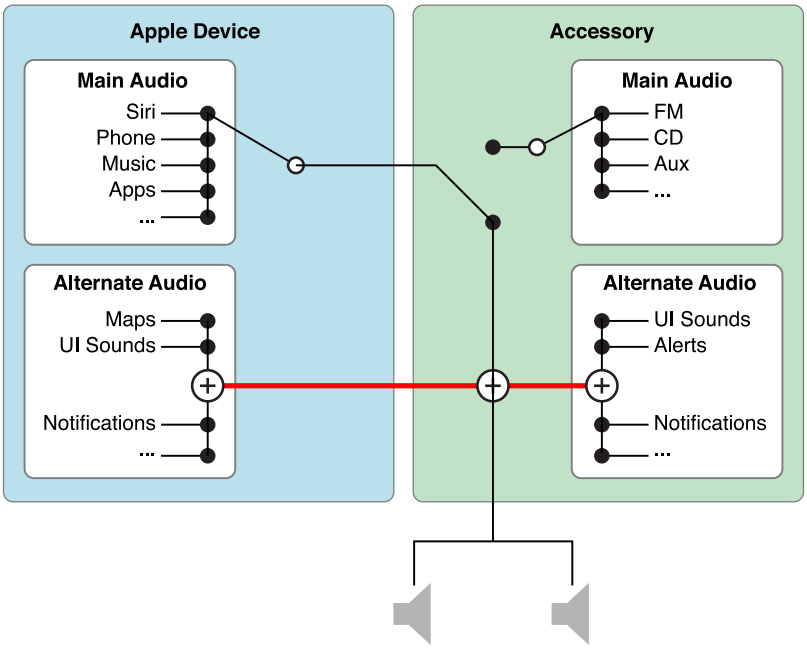
- **Screen**: The automotive head unit screen.
- **Main Audio**: The full-duplex (input and output) audio stream. There can be only one source (either accessory or Apple device) for main audio. Voice control, Siri, telephony, and media are examples of audio that uses this stream.

- **Alternate Audio**: An output-only audio stream. There can be multiple sources of alternate audio, which must be mixed together (possibly with Ducking (page 423)). UI sounds, alerts, and navigation prompts are examples of audio that uses this stream.

Screen and main audio are managed as a shared resource and both the accessory and the device must request an ownership (see Resource Ownership (page 450)), where as the alternate audio stream must be available to the device at all times.

Both accessory and Apple device must distinguish between main and alternate audio, and route audio accordingly. Figure 23-8 (page 449) illustrates conceptually the various audio sources and shows how they must be mixed.

Figure 23-8 Mixing CarPlay audio



23.3.3.2 App States

Three app-specific states must be communicated between the accessory and the Apple device:

- **Speech**: An entity (accessory or Apple device) is performing a speech operation (such as, speaking or recognizing speech). All audio output from the accessory must be disabled while speech is being recorded to prevent interference with recognition algorithms. When an entity is speaking to the user, other entities must avoid playing speech. For example, during this time navigation prompts can be replaced with tones or other indicators.
- **Phone Call**: An entity is on a phone call. Only a single entity must be in this state at a time.

- **Turn-by-turn Navigation**: An entity is performing turn-by-turn navigation. Only a single entity must be in this state at a time.

Note that additional app states may be added in future versions of CarPlay. Also note that the values of the app state properties are purely informative, to allow both the accessory and Apple device to make policy decisions that improve the user experience. For example:

- While one entity is performing speech recognition, all audio from the accessory must be silenced to avoid interfering with the speech recognition algorithms.
- While one entity is speaking to the user, other speech must be suppressed or conveyed as tones rather than speech; multiple, simultaneous speakers can be confusing to the user.
- While one entity is engaged in a phone call, the other entity cannot initiate a phone call (first caller wins).
- If one entity starts performing turn-by-turn navigation, the other entity must stop doing its turn-by-turn navigation, if applicable (last system wins).

The accessory must respect the current values of the app state properties. These properties must not be ignored unless it is essential to give critical safety or emergency information to the user.

23.3.3.3 Resource Ownership

Only one entity (accessory or Apple device) can claim ownership of a resource at any given time. When one entity claims ownership of a resource, it also constrains the circumstances under which ownership can be transferred. The three constraint modifiers are:

- **Anytime**: The other entity may take/borrow the resource at any time. If the screen is showing unimportant information, it would use this value.
- **User Initiated**: The other entity may take or borrow this resource, but doing so would disrupt the current user experience, and must only happen in response to a user-initiated action, and only if acquiring the resource is crucial to the user experience of the new mode. While music is playing, for example, the main audio channel must not be taken or borrowed unless the user has switched to a different audio source.

  A user-initiated action is defined as one where the user taps a user interface element, presses a button, turns a knob, and so forth, in order to instruct an entity to set a new context, change source, and so forth. The accessory must not initiate any 'user-initiated' actions on the user's behalf unless these are required to give critical safety or emergency information to the user.
- **Never**: The other entity may not take or borrow this resource under any circumstances. The owner/borrower of the resource would set this during a phone call or while a safety alert is being shown.

The accessory must set the borrow constraint to Anytime except for UI that requires direct user interaction or is safety related.

Ownership of a resource can be permanently transferred (taken) or temporarily transferred (borrowed).

Take: Ownership of the resource is permanently transferred to the other entity. If the user switches from the accessory's FM radio to the Apple device's Music app, for example, the Apple device would take the main audio channel from the accessory.

Borrow: Ownership is temporarily transferred. When ownership is returned, the owner should resume whatever it was doing with the resource. If the user is listening to the accessory's FM radio and starts a Siri session, for example, the Apple device would borrow the main audio channel. When the Siri session ends, ownership returns to the accessory and the FM radio must continue.

When a resource is borrowed, the other entity still retains ownership and must maintain its state. Maintaining the original ownership state (with take constraints) is important to support the following three possible ways for a borrow state to end:

- The entity borrowing the resource has finished using it and returns it to the owner. This is the typical case. For example, when a phone call ends the main audio channel may return to the accessory, which continues playing FM radio.
- The owner requires the resource and prematurely ends the borrowing state, using the take constraint of the borrow state. For example, the Apple device may borrow the main audio channel to start a Siri session, but the accessory prematurely ends the session because the main audio channel is needed for an incoming call from a phone in the vehicle that does not support CarPlay.
- The entity borrowing the resource wishes to convert from borrowing to ownership, using the original owner's take constraints. For example, the user may ask Siri to switch from FM radio to the Apple device's music player. The Siri session initially borrows the main audio channel, but when the music player starts it will want to own it.

Taking a resource with a constraint of Never is not recommended. Instead, the accessory should borrow the resource with an unborrow constraint of Never and then unborrow the resource when finished with it.

23.3.3.4 Examples of Resource Management

For examples of mode changes, please refer to the *Resource Management* document in the CarPlay DevKit.

23.3.4 Modes

Modes provide a mechanism to manage shared resources (e.g. the main screen), and state (e.g. on a phone call) during a CarPlay session. They are represented by the values in the following tables:

Table 23-51 Entity enum values

Name	Value	Description
None	0	Only used for appStates to indicate that no entity is in one of the App States. May not be used to describe ownership for a given resource (Main Screen or Main Audio).
Controller	1	The Apple device owns the given resource (Main Screen or Main Audio) or is in the given app state.
Accessory	2	The accessory owns the given resource (Main Screen or Main Audio) or is in the given app state.

Table 23-52 Resource ID enum values

Name	Value	Description
Main Screen	1	The main display for the system.
Main Audio	2	The main audio stream used for input, output, or full-duplex audio, used by telephony and media. This mode requires exclusive access; either the Apple device or the accessory may use it but not both at once.

Table 23-53 Resource constraint enum values

Name	Value	Description
Anytime	100	Resource may be taken or borrowed at any time.
User Initiated	500	Resource may be taken or borrowed if user requests.
Never	1000	Resource may never be taken or borrowed.

Table 23-54 Resource ownership transfer type enum values

Name	Value	Description
Take	1	Acquire ownership of a resource permanently. For example, if the user switches from the accessory's FM radio to the Apple device's music app, the Apple device would "take" main audio.
Untake	2	Indicate that a resource is no longer needed. Ownership of a resource does not change immediately, it changes the next time the Apple device requests it. The accessory is notified using a "modesChanged" command (see modesChanged (page 480)).

Name	Value	Description
Borrow	3	Transfer ownership temporarily. For example, if the user is listening to the accessory's FM radio and then starts a Siri session, the Apple device would "borrow" main audio. When the Siri session ends, the Apple device would "unborrow" main audio.
Unborrow	4	Release ownership of a resource that was acquired temporarily.

Table 23-55 Resource transfer priority enum values

Name	Value	Description
Nice to Have	100	Transfer succeeds only if constraint is Anytime. Must only be used with a constraint of Anytime.
User Initiated	500	Transfer succeeds only if constraint is User Initiated or Anytime.

Table 23-56 App state enum values

Name	Value	Description
Speech	1	An entity is performing a speech operation. All audio output must be disabled while speech is being recorded to prevent interference with recognition algorithms. When an entity is speaking to the user, other entities must avoid playing speech. During this time, navigation prompts, etc. can be replaced with tones or other indicators.
PhoneCall	2	An entity is engaged in a phone call. Only a single entity must be in this state at a time.
TurnByTurn	3	An entity is performing turn-by-turn navigation. Only a single entity must be in this state at a time.

Table 23-57 Speech mode enum values

Name	Value	Description
None	-1	No speech-related states are active.
Speaking	1	An entity is speaking to the user.
Recognizing Speech	2	An entity is recording audio to recognize speech from the user.

23.3.4.1 Initial Mode

When a session starts (on connection, for example), the Apple device queries the accessory for its current mode to synchronize the states of the two devices. This is done via the Info Message (page 430). The accessory must respond with its current mode, including the type, priority and constraints for each resource, as well as current app states and speech mode. The Apple device interprets this initial state as a resource transfer to the accessory and will only then evaluate whether any mode changes are required and permissible. For example, if music was already playing on the Apple device prior to connection to the accessory, the Apple device would then "take" main audio, provided that this action was compatible with the accessory's initial mode constraints.

An accessory must not take audio with a constraint of Never and should not take the screen with a constraint of Never.

Synchronization of the Initial Mode may be required beyond just the start of a CarPlay session: the accessory must be ready to respond to an Info message requesting an update of its current mode at any time.

23.3.4.2 Mode Changes

To modify the current mode, the accessory sends the Apple device a mode change request communicating the intent of the mode change. This is done via the `/command` URL (See Commands (page 477)). The Apple device takes the request and determine a new mode based on it. The Apple device will reject badly-formed requests or requests that are not compatible with the current resource constraints; in either case an error will be returned to the accessory. Once the Apple device has evaluated a new mode, it communicates it to the accessory (see Current Mode (page 456)). Mode changes originating from the Apple device are communicated directly to the accessory as an update to the current mode; they are not preceded by a mode change request from the Apple device to the accessory.

The accessory must honor any updates to the current mode within 100 ms of receiving them from the Apple device. For example, it must relinquish control of the screen within 100 ms if screen ownership is transferred to the Apple device.

A mode change request consists of the following information:

Main Screen:

- Transfer Type: Take, Untake, Borrow, or Unborrow
- Priority: Nice to Have or User Initiated
- Constraints: (1 or 2 constraints, depending on the Transfer Type)
- For Transfer Type Take, Apple device may Take back Main Screen: Anytime or User Initiated or Never
- For Transfer Type Take, Apple device may Borrow back Main Screen: Anytime or User Initiated or Never
- For Transfer Type Borrow, Apple device may Unborrow Main Screen: Anytime or User Initiated or Never

Main Audio:

- Transfer Type: Take, Untake, Borrow, or Unborrow
- Priority: Nice to Have or User Initiated
- Constraints: (1 or 2 constraints, depending on the Transfer Type)
- For Transfer Type Take, Apple device may Take back Main Audio: Anytime or User Initiated or Never
- For Transfer Type Take, Apple device may Borrow back Main Audio: Anytime or User Initiated or Never
- For Transfer Type Borrow, Apple device may Unborrow Main Audio: Anytime or User Initiated or Never

App States:

- Speech: Speaking, Speech Recognizing, or Neither
- Phone Call: True or False
- Turn-by-turn navigation: True or False

A change request must include at least one of the foregoing properties. If an action requires changes to multiple properties of a mode, they should be sent in a single transaction.

Not all mode change requests will require changes to all properties, in which case unchanged properties can be left out.

It is important that the accessory always update the Apple device with its current state, regardless of the current mode. For example, if the user switches from FM radio to the CD player, ownership of the main audio will not change. But the accessory should still send a mode change request indicating that it wants to take ownership of main audio.

The accessory must specify whether it wishes to own or borrow a resource, as described in the Resource Ownership (page 450). When the accessory has finished borrowing a resource, it must send a mode change request to release it. Each successful borrow must have a matching unborrow, for example, if an accessory borrows main audio 3 times, it must unborrow 3 times to release it to the previous owner. The release transfer type indicates to the Apple device that the accessory no longer needs the resource, and the resource constraints will be reset to "Other may take anytime". A successful taking or borrowing of a resource currently borrowed by the other entity will terminate the borrow, as described in Resource Ownership (page 450).

The accessory can use a priority value to indicate to the Apple device how badly it wants a resource. A priority of "nice to have" will succeed if the take/borrow constraint is "anytime." A priority of "user-initiated" will succeed if the take/borrow constraint is "user-initiated" or "anytime."

In the case of a non-routine safety or emergency situation (for example, a flat tire), the accessory should still send a changeModes request, but it does not have to wait for a modesChanged notification before taking over video or audio. In all other routine cases, the accessory must wait for the notification.

The accessory must be ready to handle setup or record events from the Apple device within 200 ms of the time that the accessory issues the changeModes command (see changeModes (page 479)) or receives a modesChanged command (see modesChanged (page 480)).

23.3.4.3 Current Mode

The current mode that the Apple device communicates to the accessory includes these parameters:

- **Accessory** or **Apple device** has Screen
- **Accessory** or **Apple device** has Main Audio
- **Accessory** or **Apple device** or **Neither** is performing speech recognition or speech
- **Accessory** or **Apple device** or **Neither** is engaged in a phone call
- **Accessory** or **Apple device** or **Neither** is performing turn-by-turn navigation

Note that if ownership of a resource is returned to the accessory when the accessory didn't explicitly ask for it, the accessory should resume whatever it was last doing with that resource, if possible.

23.3.5 User Input

This section covers the delivery of user input events for CarPlay and assumes familiarity with the Device Class for Human Interface Devices (HID) specification, the HID Usage Tables and addendum that are published by the USB Implementors Forum. For further information please visit http://www.usb.org/developers/hidpage/.

See additional requirements in HID Requirements (page 580).

Each HID device dictionary contains a HID descriptor (see USB HID specification), device UUID, and a display UUID directing the HID events to a specific display. Table 23-58 (page 456) lists the possible HID device description keys stored in an HID device dictionary.

Table 23-58 HID device description keys

Key	Type	Required?	Description
displayUUID	string	Y	UUID of a display associated with the HID device or a null UUID if not associated to a display.
hidCountryCode	number	Y	USB-style HID country code.

Key	Type	Required?	Description
hidDescriptor	data	Y	USB-formatted HID descriptor.
hidProductID	number	Y	USB-style HID product ID. Must be non-zero.
hidVendorID	number	Y	USB-style HID vendor ID. Must be non-zero.
name	string	Y	User-friendly name of the HID device.
uuid	string	Y	UUID to uniquely identify the HID device.

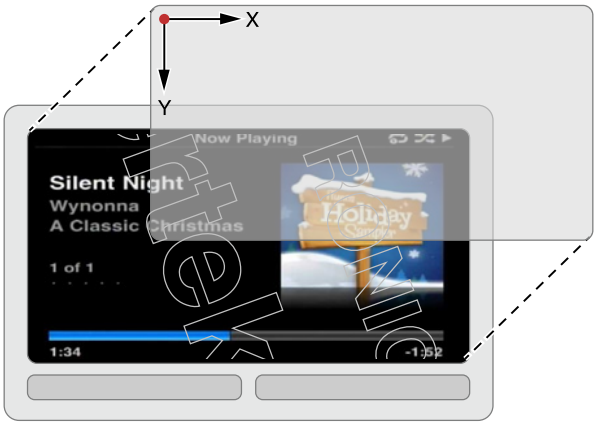
23.3.5.1 Digitizer Support (Touchscreen and Touchpad)

A digitizer is a device that measures absolute spatial position, typically in two or more dimensions. CarPlay supports the following types of digitizers:

23.3.5.1.1 Touchscreen

A touchscreen is a digitizer with an integrated display that allows the use of a finger for direct interaction with a presented interface.

Figure 23-9 Touchscreen

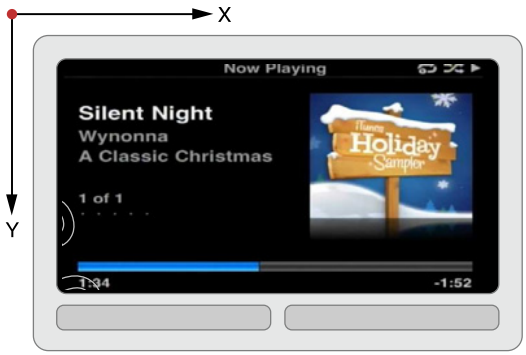


The touch surface must be sampled at the same rate as the refresh rate of the video stream on the integrated display. A sampling and refresh rate of 60 Hz is recommended. See High Resolution Display (page 383) and User Interface (UI) Stream (page 411).

23.3.5.1.2 Touchpad

A touchpad is a digitizer without an integrated display that allows the use of a finger for indirect interaction with a presented interface.

Figure 23-10 Touchpad



Generally, reports originating from this class of device would entailed relative movements synonymous with a mouse. These devices must only convey absolute transducer movement.

Accessories using a touchpad as the primary user interface must support the following:

- Single Touch as a touchpad, see Single Touch (page 459)
- Button 1 (Primary Button) to convey selection
- AC Back button

If the accessory supports character recognition, it must provide character input gesture support (see Character Input Gesture Support (page 461)) using hidSetInputMode (see hidSetInputMode (page 482)).

The accessory must use touchpad as a primary input device only with supported iOS versions.

See hidSetInputMode (page 482) to configure the input mode of a HID device and hidSendReport (page 481) for delivering HID reports.

23.3.5.1.3 HID Descriptor Support

The accessory may support the following HID usages:

Table 23-59 Digitizer Support HID Usages

Page ID	Page Name	Usage ID	Usage Name	Usage Type
0x01	Generic	0x30	X	Dynamic Value
0x01	Generic	0x31	Y	Dynamic Value
0x0D	Digitizer	0x04	Touch Screen	Application Collection

Page ID	Page Name	Usage ID	Usage Name	Usage Type
0x0D	Digitizer	0x05	Touch Pad	Application Collection
0x0D	Digitizer	0x22	Finger	Logical Collection
0x0D	Digitizer	0x32	In Range	Momentary Control
0x0D	Digitizer	0x33	Touch	Momentary Control
0x0D	Digitizer	0x37	Data Valid	Momentary Control
0x0D	Digitizer	0x38	Transducer Index	Dynamic Value
0x0D	Digitizer	0x42	Tip Switch	Momentary Control
0x0D	Digitizer	0x51	Contact Identifier	Dynamic Value

When the Apple device does not own the screen, the accessory must not send these HID usages.

23.3.5.1.4 Single Touch

Single touch implies the ability to convey the movement of only one finger over a digitizer surface.

In order to ensure proper detection, an accessory must declare an application collection using either one of the following usages:

- Touch Screen
- Touch Pad

The accessory must declare a logical collection using the following usage:

- Finger

Each logical collection must also contain absolute X and Y axes:

- X
- Y

And either one of the following to convey a touch down:

- Touch
- Tip Switch

In addition, the accessory must supply the following:

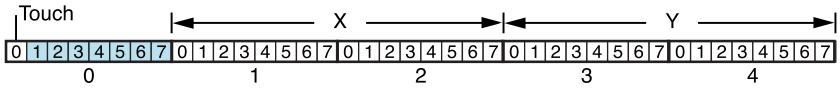
- Physical Minimum
- Physical Maximum
- Unit Exponent
- Unit

The following is an example report descriptor and format for a single-touch screen:

```
0x05, 0x0D,          // Usage Page (Digitizer)
0x09, 0x04,          // Usage (Touch Screen)
0xA1, 0x01,          // Collection (Application)
0x05, 0x0D,          //   Usage Page (Digitizer)
0x09, 0x22,          //   Usage (Finger)
0xA1, 0x02,          //   Collection (Logical)
0x05, 0x0D,          //     Usage Page (Digitizer)
0x09, 0x33,          //     Usage (Touch)
0x15, 0x00,          //     Logical Minimum......... (0)
0x25, 0x01,          //     Logical Maximum......... (1)
0x35, 0x00,          //     Physical Minimum........ (0)
0x46, 0x64, 0x00,    //     Physical Maximum........ (100)
0x55, 0x0F,          //     Unit Exponent (-1)
0x65, 0x11,          //     Unit (cm)
0x75, 0x01,          //     Report Size............. (1)
0x95, 0x01,          //     Report Count............ (1)
0x81, 0x02,          //     Input...................(Data, Variable, Absolute)
0x75, 0x07,          //     Report Size............. (7)
0x95, 0x01,          //     Report Count............ (1)
0x81, 0x01,          //     Input...................(Constant)
0x05, 0x01,          //     Usage Page (Generic Desktop)
0x09, 0x30,          //     Usage (X)
0x15, 0x00,          //     Logical Minimum......... (0)
0x26, 0x20, 0x03,    //     Logical Maximum......... (800)
0x75, 0x10,          //     Report Size............. (16)
0x95, 0x01,          //     Report Count............ (1)
```

```
0x81, 0x02,          //     Input..................(Data, Variable, Absolute)
0x09, 0x31,          //     Usage (Y)
0x15, 0x00,          //     Logical Minimum......... (0)
0x26, 0xE0, 0x01,    //     Logical Maximum......... (480)
0x75, 0x10,          //     Report Size.............. (16)
0x95, 0x01,          //     Report Count............. (1)
0x81, 0x02,          //     Input..................(Data, Variable, Absolute)
0xC0,                //   End Collection
0xC0,                // End Collection
```

Figure 23-11 Example Input Report Layout for Single-Touch Screen



23.3.5.2 Character Input Gesture Support

We are seeking to add functional enhancements to the HID Digitizer Page (0x0D) to convey the accessory's ability to process character generating gestures. Though there already exists a HID unicode page, it is limited to USC-2 (UTF16-LE). As a result, this prevents accessories from fully supporting certain Asian character sets.

We are proposing the addition of support for the transmitting of character strings with alternate encodings such as UTF8, UTF16 and UTF32.

The accessory may support the following HID usages:

Table 23-60 Character Input Gesture Support HID Usages

Page ID	Page Name	Usage ID	Usage Name	Usage Type
0x0D	Digitizer	0x23	Device Settings	Logical Collection
0x0D	Digitizer	0x24	Character Gesture	Logical Collection
0x0D	Digitizer	0x60	Character Gesture Enable	Dynamic Flag
0x0D	Digitizer	0x61	Character Gesture Quality	Dynamic Value
0x0D	Digitizer	0x62	Character Gesture Data Length	Buffered Bytes
0x0D	Digitizer	0x63	Character Gesture Data	Dynamic Value
0x0D	Digitizer	0x64	Character Gesture Encoding	Named Array

Page ID	Page Name	Usage ID	Usage Name	Usage Type
0x0D	Digitizer	0x65	UTF8 Character Gesture Encoding	Selector
0x0D	Digitizer	0x66	UTF16 Little Endian Character Gesture Encoding	Selector
0x0D	Digitizer	0x67	UTF16 Big Endian Character Gesture Encoding	Selector
0x0D	Digitizer	0x68	UTF32 Little Endian Character Gesture Encoding	Selector
0x0D	Digitizer	0x69	UTF32 Big Endian Character Gesture Encoding	Selector

When the Apple device does not own the screen, the accessory must not send these HID usages.

23.3.5.2.1 Basic Character Gesture Recognition

Basic character gesture recognition allows an accessory to convey a single character string as a result of interpreting transducer movement on a digitizer surface.

In order for the Apple device to properly detect support for these gestures, the accessory must declare a logical collection with the following usages:

- Character Gesture
- Character Gesture Data Length
- Character Gesture Data

Additionally, the accessory must also include string encoding information. If more than one encoding type is supported, they must be placed in a selector array. Otherwise, the accessory may declare individual encoding support via a static item. Use the following usages:

- Character Gesture Encoding
- UTF8 Character Gesture Encoding
- UTF16 Little Endian Character Gesture Encoding
- UTF16 Big Endian Character Gesture Encoding
- UTF32 Little Endian Character Gesture Encoding
- UTF32 Big Endian Character Gesture Encoding

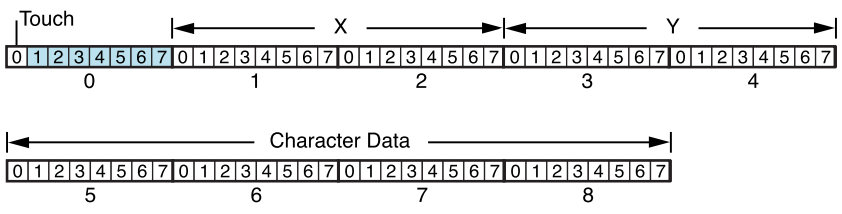
The Apple device will notify the accessory of the input mode using the hidSetInputMode as described in hidSetInputMode (page 482).

The following is an example report descriptor and format for a single-touch touchpad with basic character gesture recognition:

```
0x05, 0x0D,          // Usage Page (Digitizer)
0x09, 0x05,          // Usage (Touch Pad)
0xA1, 0x01,          // Collection (Application)
0x05, 0x0D,          //   Usage Page (Digitizer)
0x09, 0x22,          //   Usage (Finger)
0xA1, 0x02,          //   Collection (Logical)
0x05, 0x0D,          //     Usage Page (Digitizer)
0x09, 0x33,          //     Usage (Touch)
0x15, 0x00,          //     Logical Minimum......... (0)
0x25, 0x01,          //     Logical Maximum......... (1)
0x35, 0x00,          //     Physical Minimum........ (0)
0x46, 0x64, 0x00,    //     Physical Maximum........ (100)
0x55, 0x0F,          //     Unit Exponent (-1)
0x65, 0x11,          //     Unit (cm)
0x75, 0x01,          //     Report Size............. (1)
0x95, 0x01,          //     Report Count............ (1)
0x81, 0x02,          //     Input...................(Data, Variable, Absolute)
0x75, 0x07,          //     Report Size............. (7)
0x95, 0x01,          //     Report Count............ (1)
0x81, 0x01,          //     Input...................(Constant)
0x05, 0x01,          //     Usage Page (Generic Desktop)
0x09, 0x30,          //     Usage (X)
0x15, 0x00,          //     Logical Minimum......... (0)
0x26, 0x00, 0x04,    //     Logical Maximum......... (1024)
0x35, 0x00,          //     Physical Minimum........ (0)
0x46, 0x64, 0x00,    //     Physical Maximum........ (100)
0x55, 0x0F,          //     Unit Exponent (-1)
0x65, 0x11,          //     Unit (cm)
0x75, 0x10,          //     Report Size............. (16)
0x95, 0x01,          //     Report Count............ (1)
```

```
0x81, 0x02,          //     Input...................(Data, Variable, Absolute)
0x09, 0x31,          //     Usage (Y)
0x15, 0x00,          //     Logical Minimum......... (0)
0x26, 0x00, 0x04,    //     Logical Maximum......... (1024)
0x35, 0x00,          //     Physical Minimum........ (0)
0x46, 0x64, 0x00,    //     Physical Maximum........ (100)
0x55, 0x0F,          //     Unit Exponent (-1)
0x65, 0x11,          //     Unit (cm)
0x75, 0x10,          //     Report Size.............. (16)
0x95, 0x01,          //     Report Count............. (1)
0x81, 0x02,          //     Input...................(Data, Variable, Absolute)
0xC0,                //   End Collection
0x05, 0x0D,          //   Usage Page (Digitizer)
0x09, 0x24,          //   Usage (Gesture Character)
0xA1, 0x02,          //   Collection (Logical)
0x05, 0x0D,          //     Usage Page (Digitizer)
0x09, 0x63,          //     Usage (Gesture Character Data)
0x75, 0x20,          //     Report Size.............. (32)
0x95, 0x01,          //     Report Count............. (1)
0x82, 0x02, 0x01,    //     Input...................(Data, Variable, Absolute,
Buffered bytes)
0x09, 0x65,          //     Usage (Gesture Character Encoding UTF8)
0x09, 0x62,          //     Usage (Gesture Character Data Length)
0x75, 0x08,          //     Report Size.............. (8)
0x95, 0x02,          //     Report Count............. (2)
0x81, 0x02,          //     Input...................(Data, Variable, Absolute)
0xC0,                //   End Collection
0xC0,                // End Collection
```

Figure 23-12 Example Input Report Layout for Single-Touch Touchpad with Basic Character Gesture Recognition



When reporting character data, care must taken to ensure proper processing of repeated characters. This must be accomplished by issuing a subsequent report that clears gesture data and data length for a given character collection. Using the report format from the example above, the following sequence illustrates how this is accomplished ignoring current finger touch and location states:

First HID report dispatched containing gesture event for the character 'A':

```
XX XX XX XX XX 41 00 00 00 01 65
```

Second HID report dispatched clearing the gesture:

```
XX XX XX XX XX 00 00 00 00 00 65
```

23.3.5.2.2 Character Gesture Recognition with Alternate Interpretations

There can be situations in which the recognition system generates more than one interpretation of a gesture motion. We are proposing the ability to convey alternate gesture interpretations from a single accessory which will allow the Apple device to select the appropriate string based on its current application context.

Each gesture item follows the requirements detailed in Basic Character Gesture Recognition, but must also include a declaration for Character Gesture Quality in each logical collection. This will give the Apple device additional qualitative information so that it can select the appropriate interpretation.

In addition, if no alternative interpretations are available, the recognition system must inform the Apple device by ensuring that only the first character is populated and all subsequent characters are cleared.

The following is an example report descriptor and format for a single-touch touchpad with character gesture recognition with alternate interpretations:

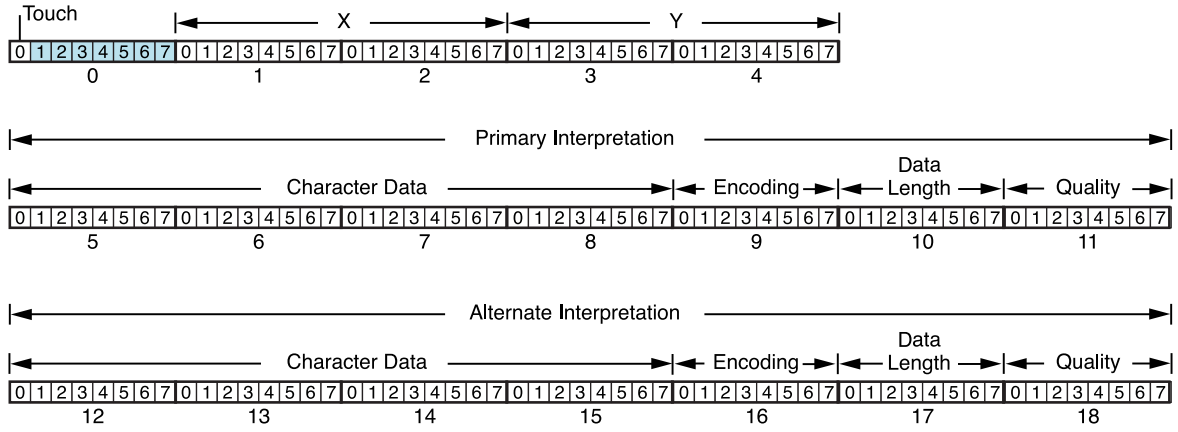
```
0x05, 0x0D,          // Usage Page (Digitizer)
0x09, 0x05,          // Usage (Touch Pad)
0xA1, 0x01,          // Collection (Application)
0x05, 0x0D,          //   Usage Page (Digitizer)
0x09, 0x22,          //   Usage (Finger)
0xA1, 0x02,          //   Collection (Logical)
0x05, 0x0D,          //     Usage Page (Digitizer)
0x09, 0x33,          //     Usage (Touch)
0x15, 0x00,          //     Logical Minimum......... (0)
0x25, 0x01,          //     Logical Maximum......... (1)
0x35, 0x00,          //     Physical Minimum........ (0)
```

```
0x46, 0x64, 0x00,    //     Physical Maximum........ (100)
0x55, 0x0F,          //     Unit Exponent (-1)
0x65, 0x11,          //     Unit (cm)
0x75, 0x01,          //     Report Size............. (1)
0x95, 0x01,          //     Report Count............ (1)
0x81, 0x02,          //     Input...................(Data, Variable, Absolute)
0x75, 0x07,          //     Report Size............. (7)
0x95, 0x01,          //     Report Count............ (1)
0x81, 0x01,          //     Input...................(Constant)
0x05, 0x01,          //     Usage Page (Generic Desktop)
0x09, 0x30,          //     Usage (X)
0x15, 0x00,          //     Logical Minimum......... (0)
0x26, 0x00, 0x04,    //     Logical Maximum......... (1024)
0x35, 0x00,          //     Physical Minimum........ (0)
0x46, 0x64, 0x00,    //     Physical Maximum........ (100)
0x55, 0x0F,          //     Unit Exponent (-1)
0x65, 0x11,          //     Unit (cm)
0x75, 0x10,          //     Report Size............. (16)
0x95, 0x01,          //     Report Count............ (1)
0x81, 0x02,          //     Input...................(Data, Variable, Absolute)
0x09, 0x31,          //     Usage (Y)
0x15, 0x00,          //     Logical Minimum......... (0)
0x26, 0x00, 0x04,    //     Logical Maximum......... (1024)
0x35, 0x00,          //     Physical Minimum........ (0)
0x46, 0x64, 0x00,    //     Physical Maximum........ (100)
0x55, 0x0F,          //     Unit Exponent (-1)
0x65, 0x11,          //     Unit (cm)
0x75, 0x10,          //     Report Size............. (16)
0x95, 0x01,          //     Report Count............ (1)
0x81, 0x02,          //     Input...................(Data, Variable, Absolute)
0xC0,                //   End Collection
0x05, 0x0D,          //   Usage Page (Digitizer)
0x09, 0x24,          //   Usage (Gesture Character)
0xA1, 0x02,          //   Collection (Logical)
```

```
0x05, 0x0D,          //     Usage Page (Digitizer)
0x09, 0x63,          //     Usage (Gesture Character Data)
0x75, 0x20,          //     Report Size..... (32)
0x95, 0x01,          //     Report Count..... (1)
0x82, 0x02, 0x01,    //     Input...................(Data, Variable, Absolute,
Buffered bytes)
0x09, 0x65,          //     Usage (Gesture Character Encoding UTF8)
0x09, 0x62,          //     Usage (Gesture Character Data Length)
0x75, 0x08,          //     Report Size..... (8)
0x95, 0x02,          //     Report Count..... (2)
0x81, 0x02,          //     Input...................(Data, Variable, Absolute)
0x09, 0x61,          //     Usage (Gesture Character Quality)
0x15, 0x00,          //     Logical Minimum......... (0)
0x25, 0x64,          //     Logical Maximum......... (100)
0x75, 0x08,          //     Report Size..... (8)
0x95, 0x01,          //     Report Count..... (1)
0x81, 0x02,          //     Input...................(Data, Variable, Absolute)
0xC0,                //   End Collection
0x05, 0x0D,          //   Usage Page (Digitizer)
0x09, 0x24,          //   Usage (Gesture Character)
0xA1, 0x02,          //   Collection (Logical)
0x05, 0x0D,          //     Usage Page (Digitizer)
0x09, 0x63,          //     Usage (Gesture Character Data)
0x75, 0x20,          //     Report Size..... (32)
0x95, 0x01,          //     Report Count..... (1)
0x82, 0x02, 0x01,    //     Input...................(Data, Variable, Absolute,
Buffered bytes)
0x09, 0x65,          //     Usage (Gesture Character Encoding UTF8)
0x09, 0x62,          //     Usage (Gesture Character Data Length)
0x75, 0x08,          //     Report Size..... (8)
0x95, 0x02,          //     Report Count..... (2)
0x81, 0x02,          //     Input...................(Data, Variable, Absolute)
0x09, 0x61,          //     Usage (Gesture Character Quality)
0x15, 0x00,          //     Logical Minimum......... (0)
0x25, 0x64,          //     Logical Maximum......... (100)
0x75, 0x08,          //     Report Size..... (8)
```

```
0x95, 0x01,          //     Report Count..... (1)
0x81, 0x02,          //     Input.................(Data, Variable, Absolute)
0xC0,                //   End Collection
0xC0,                // End Collection
```

Figure 23-13 Example Input Report Layout for Single-Touch Touchpad with Character Gesture Recognition with Alternate Interpretations

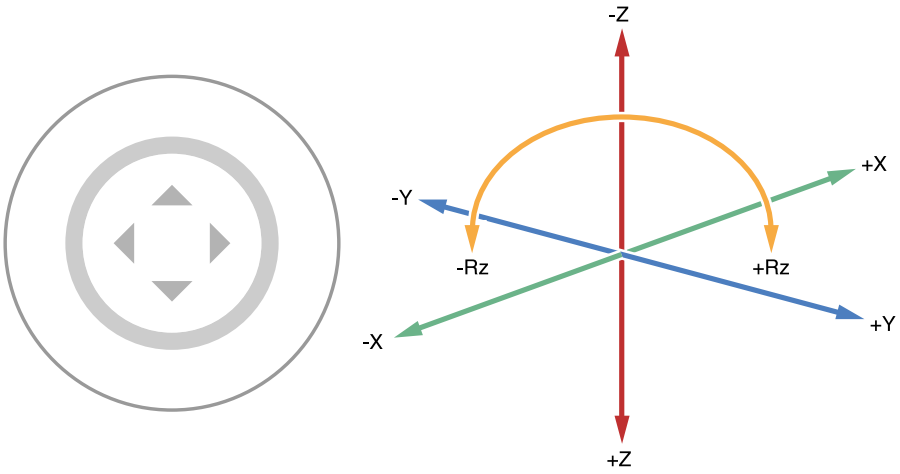


23.3.5.3 Knob Support (Multi-axis Controller)

Knobs refer to user input devices, commonly found on a vehicle center console, which allow for indirect interaction with a media display.

A multi-axis controller is similar to a joystick but minimally consists of three variable axes (X, Y and Z). Additionally, rotational axes for X, Y, and Z may also be included.

Figure 23-14 Multi-axis Controller



The accessory may support the following HID usages:

Table 23-61 Knob Support HID Usages

Page ID	Page Name	Usage ID	Usage Name	Usage Type
0x01	Generic	0x30	X	Dynamic Value
0x01	Generic	0x31	Y	Dynamic Value
0x01	Generic	0x32	Z	Dynamic Value
0x01	Generic	0x35	Rz	Dynamic Value
0x01	Generic	0x37	Dial	Dynamic Value
0x01	Generic	0x38	Wheel	Dynamic Value
0x09	Button	0x01	Button 1 (Primary Button)	(see USB HID)
0x0D	Consumer	0x224	AC Back	One Shot Control

When the Apple device does not own the screen, the accessory must not send these HID usages.

The preferred state of the axes should signify that the knob rests at the center when not in use.

The accessory must support the following to convey translational movement:

- X
- Y

The accessory must support either one of the following to convey selection:

- Z
- Button 1 (Primary Button)

When implementing the Z axis, the Apple device only requires movement in the +Z (down) direction and will translated it to a primary button.

The accessory must support either one of the following to allow scrolling:

- Rz
- Dial
- Wheel

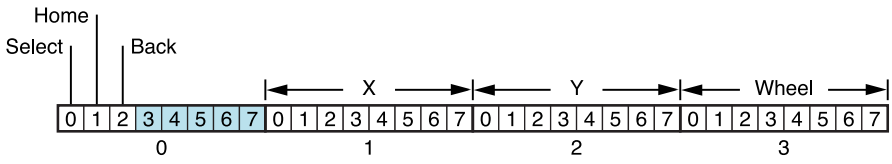
The accessory must support the AC Back button.

The following is an example report descriptor and format for a multi-axis controller:

```
0x05, 0x01,          // Usage Page (Generic Desktop)
0x09, 0x08,          // Usage (MultiAxisController)
0xA1, 0x01,          // Collection (Application)
0x05, 0x09,          //   Usage Page (Button)
0x09, 0x01,          //   Usage 1 (0x1)
0x15, 0x00,          //   Logical Minimum......... (0)
0x25, 0x01,          //   Logical Maximum......... (1)
0x75, 0x01,          //   Report Size............. (1)
0x95, 0x01,          //   Report Count............ (1)
0x81, 0x02,          //   Input...................(Data, Variable, Absolute)
0x05, 0x0C,          //   Usage Page (Consumer)
0x0A, 0x23, 0x02,    //   Usage 547 (AC Home)
0x0A, 0x24, 0x02,    //   Usage 548 (AC Back)
0x15, 0x00,          //   Logical Minimum......... (0)
0x25, 0x01,          //   Logical Maximum......... (1)
0x75, 0x01,          //   Report Size............. (1)
0x95, 0x02,          //   Report Count............ (2)
0x81, 0x02,          //   Input...................(Data, Variable, Absolute)
0x75, 0x05,          //   Report Size............. (5)
```

```
0x95, 0x01,          //   Report Count..... (1)
0x81, 0x01,          //   Input.....(Constant)
0x05, 0x01,          //   Usage Page (Generic Desktop)
0x09, 0x30,          //   Usage (X)
0x09, 0x31,          //   Usage (Y)
0x15, 0x81,          //   Logical Minimum..... (-127)
0x25, 0x7F,          //   Logical Maximum..... (127)
0x75, 0x08,          //   Report Size..... (8)
0x95, 0x02,          //   Report Count..... (2)
0x81, 0x02,          //   Input.....(Data, Variable, Absolute)
0x05, 0x01,          //   Usage Page (Generic Desktop)
0x09, 0x38,          //   Usage (Wheel)
0x15, 0x81,          //   Logical Minimum..... (-127)
0x25, 0x7F,          //   Logical Maximum..... (127)
0x75, 0x08,          //   Report Size..... (8)
0x95, 0x01,          //   Report Count..... (1)
0x81, 0x06,          //   Input.....(Data, Variable, Relative)
0xC0,                // End Collection
```

Figure 23-15 Example Input Report Layout for Multi-Axis Controller



23.3.5.4 Buttons

The CarPlay feature supports various buttons commonly found within the center console or steering wheel of a vehicle.

The accessory may support the following HID usages:

Table 23-62 Button Support HID Usages

Page ID	Page Name	Usage ID	Usage Name	Usage Type	Apple Function
0x0C	Consumer	0xB0	Play	On/Off Control	Play

Page ID	Page Name	Usage ID	Usage Name	Usage Type	Apple Function
0x0C	Consumer	0xB1	Pause	On/Off Control	Pause
0x0C	Consumer	0xB5	Scan Next Track	One Shot Control	Scan Next Track
0x0C	Consumer	0xB6	Scan Previous Track	One Shot Control	Scan Previous Track
0x0C	Consumer	0xCD	Play/Pause	One Shot Control	Toggle Play/Pause
0x0C	Consumer	0x223	AC Home	One Shot Control	CarPlay
0x0C	Consumer	0x224	AC Back	One Shot Control	Back
0x07	Keyboard	0x2A	Keyboard Delete (Backspace)	One Shot Control	Clear
0x0B	Telephony	0x20	Hook Switch	On/Off Control	Accept call
0x0B	Telephony	0x21	Flash	Momentary Control	Toggle Accept or Reject/End call
0x0B	Telephony	0x26	Drop	One Shot Control	Reject/End call
0x0B	Telephony	0x2F	Phone Mute	One Shot Control	Mute the microphone
0x0B	Telephony	0xB0	Phone Key 0	Selector	Phone Key 0
0x0B	Telephony	0xB1	Phone Key 1	Selector	Phone Key 1
0x0B	Telephony	0xB2	Phone Key 2	Selector	Phone Key 2
0x0B	Telephony	0xB3	Phone Key 3	Selector	Phone Key 3
0x0B	Telephony	0xB4	Phone Key 4	Selector	Phone Key 4
0x0B	Telephony	0xB5	Phone Key 5	Selector	Phone Key 5
0x0B	Telephony	0xB6	Phone Key 6	Selector	Phone Key 6

Page ID	Page Name	Usage ID	Usage Name	Usage Type	Apple Function
0x0B	Telephony	0xB7	Phone Key 7	Selector	Phone Key 7
0x0B	Telephony	0xB8	Phone Key 8	Selector	Phone Key 8
0x0B	Telephony	0xB9	Phone Key 9	Selector	Phone Key 9
0x0B	Telephony	0xBA	Phone Key Star	Selector	Phone Key Star
0x0B	Telephony	0xBB	Phone Key Pound	Selector	Phone Key Pound

Accessories with non-touch UIs must implement AC Back.

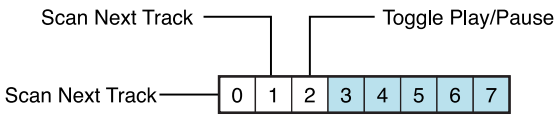
Accessories must implement one of the following:

- Flash, if the accessory has a single telephony button.
- Hook Switch & Drop, if the accessory has separate telephony accept and reject buttons.

The following is an example report descriptor and format for a simple media buttons accessory:

```
0x05, 0x0C,          // Usage Page (Consumer)
0x09, 0x01,          // Usage 1 (0x1)
0xA1, 0x01,          // Collection (Application)
0x05, 0x0C,          //   Usage Page (Consumer)
0x09, 0xB5,          //   Usage 181 (Scan Next Track)
0x09, 0xB6,          //   Usage 182 (Scan Previous Track)
0x09, 0xCD,          //   Usage 205 (Toggle Play / Pause)
0x15, 0x00,          //   Logical Minimum......... (0)
0x25, 0x01,          //   Logical Maximum......... (1)
0x75, 0x01,          //   Report Size............. (1)
0x95, 0x03,          //   Report Count............ (3)
0x81, 0x02,          //   Input...................(Data, Variable, Absolute)
0x75, 0x05,          //   Report Size............. (5)
0x95, 0x01,          //   Report Count............ (1)
0x81, 0x01,          //   Input...................(Constant)
0xC0,                // End Collection
```

Figure 23-16 Example Input Report Layout for Simple Media Buttons

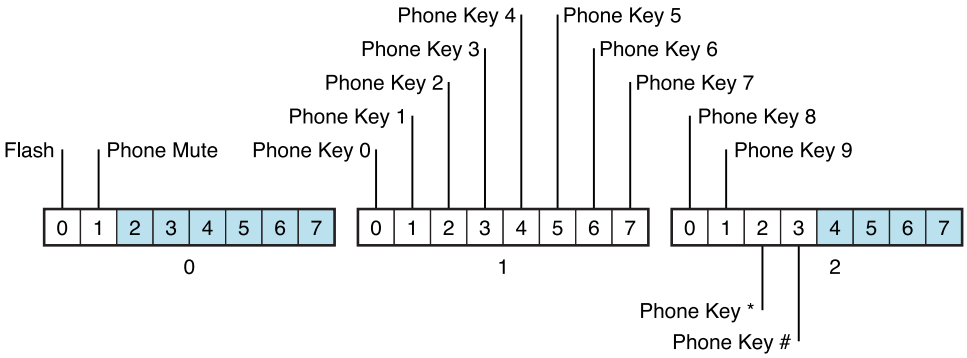


The following is an example report descriptor and format for a simple telephone buttons accessory with flash and numeric keys:

```
0x05, 0x0B,         // Usage Page (Telephony Device)
0x09, 0x01,         // Usage 1 (0x1)
0xA1, 0x01,         // Collection (Application)
0x05, 0x0B,         //   Usage Page (Telephony Device)
0x09, 0x21,         //   Usage 33 (Flash)
0x09, 0x2F,         //   Usage 47 (Phone Mute)
0x15, 0x00,         //   Logical Minimum......... (0)
0x25, 0x01,         //   Logical Maximum......... (1)
0x75, 0x01,         //   Report Size.............. (1)
0x95, 0x02,         //   Report Count............. (2)
0x81, 0x02,         //   Input...................(Data, Variable, Absolute)
0x75, 0x06,         //   Report Size.............. (6)
0x95, 0x01,         //   Report Count............. (1)
0x81, 0x01,         //   Input...................(Constant)
0x05, 0x0B,         //   Usage Page (Telephony Device)
0x09, 0xB0,         //   Usage 176 (Phone Key 0)
0x09, 0xB1,         //   Usage 177 (Phone Key 1)
0x09, 0xB2,         //   Usage 178 (Phone Key 2)
0x09, 0xB3,         //   Usage 179 (Phone Key 3)
0x09, 0xB4,         //   Usage 180 (Phone Key 4)
0x09, 0xB5,         //   Usage 181 (Phone Key 5)
0x09, 0xB6,         //   Usage 182 (Phone Key 6)
0x09, 0xB7,         //   Usage 183 (Phone Key 7)
0x09, 0xB8,         //   Usage 184 (Phone Key 8)
0x09, 0xB9,         //   Usage 185 (Phone Key 9)
0x09, 0xBA,         //   Usage 186 (Phone Key *)
0x09, 0xBB,         //   Usage 187 (Phone Key #)
```

```
0x15, 0x00,          //   Logical Minimum......... (0)
0x25, 0x01,          //   Logical Maximum......... (1)
0x75, 0x01,          //   Report Size............. (1)
0x95, 0x0C,          //   Report Count............ (12)
0x81, 0x02,          //   Input...................(Data, Variable, Absolute)
0x75, 0x04,          //   Report Size............. (4)
0x95, 0x01,          //   Report Count............ (1)
0x81, 0x01,          //   Input...................(Constant)
0xC0,                // End Collection
```

Figure 23-17 Example Input Report Layout for Simple Telephone Buttons



23.3.6 Siri User Input

Accessories may support the ability for a user to activate Siri. This is communicated to the Apple device via the requestSiri command (see requestSiri (page 482)), with the parameter being one of the following Siri actions:

Table 23-63 Siri actions enum values

Name	Value	Description
Prewarm	1	Indicate that the Apple device should begin preparing Siri. At this point, no audio or video resources will be taken.
ButtonDown	2	Indicate to the Apple device that the Siri button has been depressed.
ButtonUp	3	Indicate to the Apple device that the Siri button has been released.

Accessories may wish to overload their existing "speech recognition" button, where a short press activates the accessory's native speech recognition system, while a longer press activates Siri. The recommended value for the longer press timeout is 600 ms, but it must not be greater than 1000 ms. It is important that the accessory prewarm Siri as early as possible (for example, when the physical button is pressed) to ensure the best possible response time from Siri in the event that it is activated. After the inital Siri start up sequence, the accessory must forward all physical button presses to the device without any additional delays. Once the Siri session is finished, the accessory must return to the button's default behavior.

Short press (accessory speech recognition):

```
Physical button depressed
|        Physical button released
|        |
+--------+-------------------------------------------------------> time
|
|
Prewarm action sent
```

For short presses, where the physical button is released before the accessory-determined timeout interval, only the Prewarm action is sent to the Apple device, and the accessory's native speech recognition system is activated.

Long press (Siri):

```
Physical button depressed
|                Long Press Timeout       Physical button released
|                |                        |
+----------------+------------------------+------------------------> time
|                |                        |
|                ButtonDown action sent   ButtonUp action sent
Prewarm action sent
```

For long presses, where the physical button is held down past the accessory-determined timeout interval, the Prewarm and ButtonUp actions are sent to the Apple device, just like in the previous scenario, but when the timeout interval is reached, the accessory sends an additional ButtonDown action when the timeout interval has elapsed.

For accessories that have a dedicated Siri button, sending a Prewarm action is unnecessary and ButtonDown / ButtonUp actions are sent when the button is physically depressed and released.

23.3.7 Commands

Commands are messages exchanged between the Apple device and the accessory to perform an action. They are sent as HTTP requests using the `/command` URL on the control stream. The Apple device sends its commands over the Controller control channel while the accessory sends its commands over the Accessory control channel. Each command receives an HTTP response. The request and response payloads are binary plists. Each request contains a key, type, to indicate the command to perform. Additional keys may be included for command-specific parameters.

For example, an hidSendReport command would look like this (in XML for readability, the actual request is a binary plist):

```
<?xml version="1.0" encoding="UTF-8"?>

<!DOCTYPE plist PUBLIC "-//Apple//DTD PLIST 1.0//EN"
"http://www.apple.com/DTDs/PropertyList-1.0.dtd">

<plist version="1.0">

<dict>
    <key>hidReport</key>
    <data>
    ABEiM0Q=
    </data>
    <key>type</key>
    <string>hidSendReport</string>
    <key>uuid</key>
    <string>cbb63f31-c020-4db9-ac16-2283dda2a079</string>
</dict>
</plist>
```

23.3.7.1 duckAudio

Sender: Apple device

Description: Ramps volume down (for example, to fade down music to play a voice prompt). See Ducking (page 423). Table 23-64 (page 478) lists the specifics of duckAudio request keys.

Table 23-64 duckAudio request keys

Key	Type	Required?	Description
durationMs	number	Y	Number of milliseconds the ramp down should last.
volume	number	Y	Suggested final dB attenuation of the audio at the end of the duck (-144 to 0 dB).

23.3.7.2 unduckAudio

Sender: Apple device

Description: Ramps volume back to the pre-duck level. See Ducking (page 423). Table 23-65 (page 478) lists the specifics of unduckAudio request keys.

Table 23-65 unduckAudio request keys

Key	Type	Required?	Description
durationMs	number	Y	Number of milliseconds the ramp up should last.

23.3.7.3 disableBluetooth

Sender: Apple device

Description: Disable Bluetooth connectivity to specified device (i.e. the Apple device). If the accessory supports Bluetooth, it must provide its bluetoothIDs (see Info Message (page 430)) in order to receive a disableBluetooth command from the device. At a minimum, on receipt of this command, the accessory must immediately disable any existing connections with the specified Apple device for the following Bluetooth profiles:

- Hands-Free Profile
- Advanced Audio Distribution Profile
- A/V Remote Control Profile

In addition, the accessory must not attempt to connect any of the above-listed Bluetooth profiles with the same Apple device while the two are already connected over CarPlay (that is, a session is already in progress). Table 23-66 (page 479) lists the specifics of disableBluetooth request keys.

Table 23-66 disableBluetooth request keys

Key	Type	Required?	Description
deviceID	string	Y	MAC address of Bluetooth device to disable connectivity to (that is, the Apple device).

23.3.7.4 changeModes

Sender: Accessory

Description: Change resource states and/or app states. See Modes (page 451). Table 23-67 (page 479) lists the specifics of changeModes request keys and Table 23-70 (page 480) lists the specifics of changeModes response keys.

Table 23-67 changeModes request keys

Key	Type	Required?	Description
appStates	array	N	Application state(s) to change. Contains dictionaries of Table 23-68 (page 479).
resources	array	N	Resource ownership(s) to change. Contains dictionaries of Table 23-69 (page 479).
reasonStr	string	N	A textual description of the reason for the mode change.

Table 23-68 appState keys

Key	Type	Required?	Description
appStateID	enum	Y	One of Table 23-56 (page 453).
speechMode	enum	N (*)	(*) Required if appStateID is Speech or PhoneCall. One of Table 23-57 (page 453).
state	boolean	N (*)	(*) Required if appStateID is PhoneCall or TurnByTurn.

Table 23-69 resource keys

Key	Type	Required?	Description
resourceID	enum	Y	One of Table 23-52 (page 452).
transferType	enum	Y	One of Table 23-54 (page 452).

Key	Type	Required?	Description
transferPriority	enum	N (*)	(*) Required if transferType is take or borrow. One of Table 23-55 (page 453).
takeConstraint	enum	N (*)	(*) Required if transferType is take. One of Table 23-53 (page 452).
borrowConstraint	enum	N (*)	(*) Required if transferType is take. One of Table 23-53 (page 452).
unborrowConstraint	enum	N (*)	(*) Required if transferType is borrow. One of Table 23-53 (page 452).

Table 23-70 changeModes response keys

Key	Type	Required?	Description
params	group	N	Updated modes if the mode change was successful, otherwise absent. The modes dictionary uses the same keys as the "changeModes" command.
status	number	Y	Result of performing the mode change. If the value is nonzero, the change failed.

23.3.7.5 modesChanged

Sender: Apple device

Description: Updates the accessory's current mode after a mode change by the Apple device. Table 23-71 (page 480) lists the specifics of modesChanged request keys.

Table 23-71 modesChanged request keys

Key	Type	Required?	Description
appStates	array	N	Application state(s) to change. Contains dictionaries of Table 23-72 (page 481).
resources	array	N	Resource ownership(s) to change. Contains dictionaries of Table 23-73 (page 481).

Table 23-72 appState keys

Key	Type	Required?	Description
appStateID	enum	Y	One of Table 23-56 (page 453).
entity	enum	Y	One of Table 23-51 (page 452).
speechMode	enum	N (*)	(*) Required if appStateID is Speech or PhoneCall. One of Table 23-57 (page 453).

Table 23-73 resource keys

Key	Type	Required?	Description
resourceID	enum	Y	One of Table 23-52 (page 452).
entity	enum	Y	Value must not be 'None'. One of Table 23-51 (page 452).

23.3.7.6 forceKeyFrame

Sender: Accessory

Description: The accessory may force a key frame to be sent for the screen stream. This should only be used to recover from a decoder problem and must not be sent periodically during normal operation.

23.3.7.7 hidSendReport

Sender: Accessory

Description: When an HID event occurs on the accessory, such as the user turning a knob, it sends it to the Apple device over the control stream by sending a command with a type of hidSendReport to the `/command` URL. The command contains the USB HID report and UUID of the HID device. Table 23-74 (page 481) lists the HID event keys.

Table 23-74 hidSendReport request keys

Key	Type	Required?	Description
hidReport	data	Y	USB-formatted HID report.
timestamp	number	N	NTP timestamp when the event occurred (synchronized to the Apple device's clock).
uuid	string	Y	UUID to uniquely identify the HID device.

23.3.7.8 hidSetInputMode

Sender: Apple device

Description: Sets input mode on a HID.

Table 23-75 hidSetInputMode request keys

Key	Type	Required?	Description
hidInputMode	enum	Y	One of Table 23-76 (page 482).
uuid	string	Y	UUID to uniquely identify the HID device.

Table 23-76 HID input modes enum values

Name	Value	Description
Default	0	Default mode for non-character input (e.g. panning). See Single Touch (page 459).
Character	1	Optimize for entering characters. See Single Touch (page 459) and Character Input Gesture Support (page 461).
Scrolling	2	Optimize for non-character input (e.g. panning). See Single Touch (page 459).
ScrollingWithCharacters	3	Optimize for non-character input (e.g. panning) and entering characters. See Single Touch (page 459) and Character Input Gesture Support (page 461).
DialPad	4	Optimize for non-character input (e.g. panning) and entering dial-pad character events only (i.e., 0-9, #, *). See Single Touch (page 459) and Character Input Gesture Support (page 461), using numerical characters only.

23.3.7.9 requestSiri

Sender: Accessory

Description: Requests that Siri be invoked with a specified action.

The requestSiri command must only be sent as a result of direct user action on a physical or virtual control surface. The behavior should match that of hidSendReport (page 481), see HID Requirements (page 580).

Table 23-77 requestSiri request keys

Key	Type	Required?	Description
siriAction	enum	Y	One of Table 23-63 (page 475).

23.3.7.10 requestUI

Sender: Either Apple device or accessory

Description: For the Apple device to ask for the accessory UI to be shown, or for the accessory to ask for the CarPlay UI to be shown. The accessory must not send requestUI upon connection; the Apple device will send requestUI, if appropriate.

The requestUI command must only be sent as a result of direct user action on a physical or virtual control surface. The accessory must not send requestUI upon connection; the Apple device will request resources based on the modes provided in the Info message. See HID Requirements (page 580).

Where the sender is the accessory, a URL request key (see Table 23-78 (page 483)) is available to specify the desired CarPlay compatible app to be shown.

> **Note:** Only URLs that do not require user interaction on the Apple device are allowed.

Table 23-78 Accessory to Apple device requestUI request keys

Key	Type	Required?	Description
url	string	N	URL identifier of the desired CarPlay UI application to launch: • no url - the CarPlay screen will be shown. • "maps:" - the CarPlay Maps application will be shown. • "mobilephone:" - the CarPlay Phone application will be shown. • "music:" - the CarPlay Music application will be shown. • "nowplaying:" - the Now Playing screen will be shown. • "tel:xxx-xxx-xxxx" - the CarPlay Phone application will be shown and a phone call with the desired number will be placed (for support phone number formats see *IETF RFC 3966*.

Where the sender is the Apple device, a URL request key (see Table 23-79 (page 484)) is available to specify the desired accessory UI to be shown. This is only sent when the accessory supports the `enhancedRequestCarUI` `extendedFeatures`. See Table 23-33 (page 439).

Table 23-79 Apple device to accessory requestUI request keys

Key	Type	Required?	Description
url	string	N	URL identifier of the desired accessory UI to show: • no url - show an accessory screen which allows for re-entering the CarPlay UI. • "oem:back" - show the last accessory screen that was visible before showing the CarPlay UI.

23.3.7.11 setNightMode

Sender: Accessory

Description: The accessory may indicate whether it is dark outside.

Table 23-80 setNightMode request keys

Key	Type	Required?	Description
nightMode	boolean	Y	True if it is dark outside, false otherwise.

23.3.7.12 setLimitedUI

Sender: Accessory

Description: The accessory may indicate whether or not to limit certain UI elements. See Table 23-34 (page 439).

Table 23-81 setLimitedUI request keys

Key	Type	Required?	Description
limitedUI	boolean	Y	True if certain UI elements should be limited.

23.3.7.13 iAPSendMessage

Sender: Either Apple device or accessory

Description: Sends an iAP protocol message. This command must only be used for CarPlay over wireless sessions.

Table 23-82 iAPSendMessage request keys

Key	Type	Required?	Description
data	data	Y	The content of the iAP message.

23.3.8 CarPlay Communication Plug-in

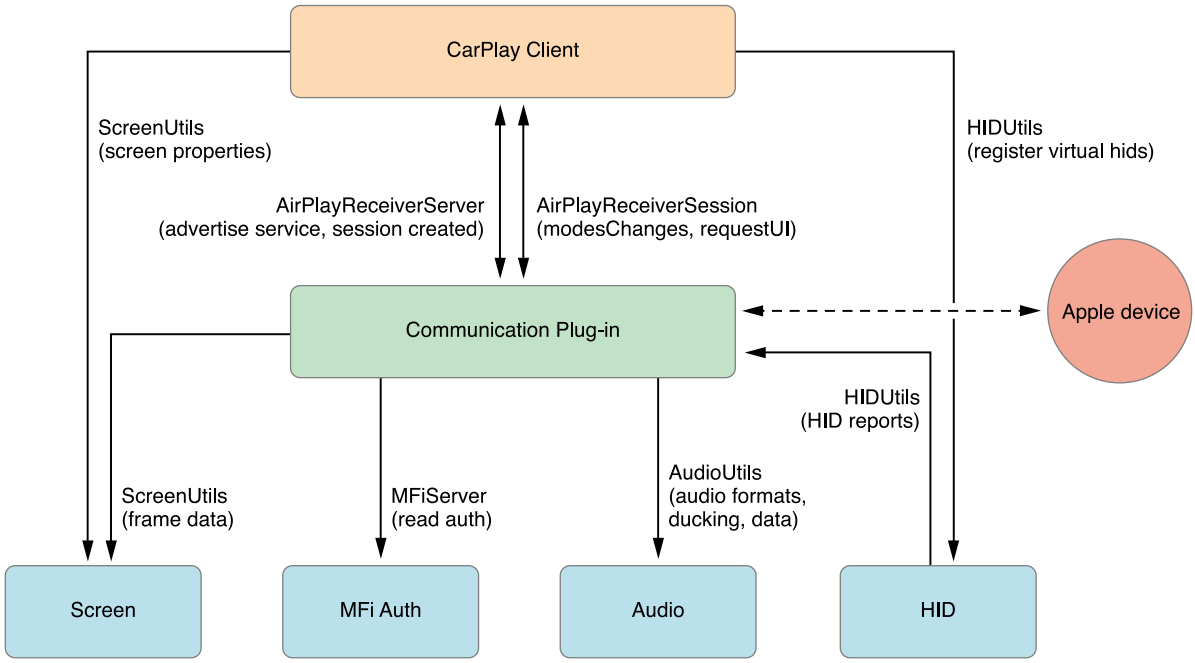
The CarPlay Communication Plug-in, source code available to accessory developers with approved CarPlay product plans, implements the core functionality described in the preceding sections of CarPlay Communication Protocol (page 424). It interfaces with Bonjour to configure and advertise the CarPlay service and establishes the communication and data links between the accessory and Apple device.

The accessory maker is not required to implement their own TCP and UDP channels for either setup and control or audio and UI transfer. Instead, these are automatically created by the Communication Plug-in which also manages authentication and encryption-decryption of all CarPlay channels.

For simplicity, the core info, setup and feedback messages are abstracted into a platform API, as they are any commands and user event. Figure 23-18 (page 486) shows an architectural overview of the CarPlay Communication Plug-in. To integrate the plug-in on a specific platform, a set of utility functions has been provided:

- **MFiServer** provides integration with platform specific installation of an MFi authentication chip, e.g. over I2C bus. The plug-in will automatically read out authentication data for encrypting the audio and video streams.
- **ScreenUtils** provides integration with the platform specific video decoding interfaces. During setup, a CarPlay client application can configure the specific screen properties. The plug-in will then directly send video frames through the provided custom APIs.
- **AudioUtils** provides integration with the platform specific audio interfaces. On each audio stream setup the plug-in will provide information about the audio formats and data, as well as forward requests to duck the current audio playback.
- **HIDUtils** provides integration with the platform specific HID devices (touchscreens, knob-based controls, etc.). A CarPlay client application can first register the available HID devices and then post HID reports to the plug-in.
- **AirPlayReceiverServer** / **AirPlayReceiverServerDelegate** - A CarPlay client application can use the delegate to send and receive server-level commands to the device, as well as obtain a reference to an active AirPlayReceiverServerSession.
- **AirPlayReceiverServerSession** / **AirPlayReceiverServerSessionDelegate** - A CarPlay client application can use the delegate to send and receive session-level commands to the device. Typical examples will be notifications for modeChanged, or commands as requestUI.

Figure 23-18 CarPlay Communication Plug-in Reference Architecture



See the technical notes provided with the CarPlay Communication Plug-in for more details on this platform abstraction and use cases.

23.4 Test Procedures

Test procedures for self certification are available for accessory manufacturers with approved product plans.