Accessory Interface Specification CarPlay Addendum

R6 Developer Preview

Developer

Contents

List of Figures

List of Tables

1 Introduction

1.1 Purpose of This Specification

This specification details requirements and recommendations for CarPlay accessories and related features.

This specification is an extension of the *Accessory Interface Specification*, therefore all requirements in that specification must be met. Chapters in this specification supersede or expand on chapters found in the *Accessory Interface Specification*.

Any conflicts between the *Accessory Interface Specification* and this specification must be resolved in favor of this specification.

1.2 Requirements, Recommendations, and Permissions

This specification contains statements that are incorporated by reference into legal agreements between Apple and its licensees. The use of the words *must*, *must not*, *required*, *shall*, *shall not*, *should*, *should not*, *recommended*, *not recommended*, *may*, *optional*, and *deprecated* in a statement have the following meanings:

- *must*, *shall*, or *required* means the statement is an absolute requirement.
- *must not*, *shall not* or *prohibited* means the statement is an absolute prohibition.
- *should* or *recommended* means the full implications must be understood before choosing a different course.
- *should not* or *not recommended* means the full implications must be understood before choosing this course.
- *may* or *optional* means the statement is truly optional, and its presence or absence cannot be assumed.
- *deprecated* means the statement is provided for historical purposes only and is equivalent to 'must not'.

The absence of requirements, recommendations, or permissions for a specific accessory design in this specification must not be interpreted as implied approval of that design. Developers are strongly encouraged to ask Apple for feedback on accessory designs that are not explicitly mentioned in this specification.

1.3 Terminology

1.3.1 Device

Device and *iOS device* refer to an iPhone, iPad, or iPod running iOS.

Where appropriate, references to specific Apple products and operating systems will also be used.

1.3.2 Accessory

In the context of CarPlay, the term *accessory* is used to refer to a vehicle and its head unit or to an aftermarket head unit. The accessory is essentially any product that connects to a *device* via the interfaces described in this specification.

1.4 Developer Preview

Content labeled as DEVELOPER PREVIEW is not intended for use in the development of Proposed Products or Licensed Products under an MFi License. Although content labeled this way has been reviewed for accuracy, it is not final. Apple is supplying this content to help accessory developers plan for the adoption of the accessory interface features described herein.

This information is subject to change, and accessories implemented according to this content must be tested with final operating system software and final documentation before going through self-certification.

2 Accessory Wi-Fi Information Sharing

Deprecated: Accessory Wi-Fi Information Sharing is deprecated and replaced by "4 CarPlay Connection" (page 180)

Accessories may share Wi-Fi configuration information with a device. This feature enables a user to quickly join a Wi-Fi network by connecting or pairing the device to an accessory over a wired or Bluetooth transport. The accessory can then notify the device of the Wi-Fi configuration information of an available Wi-Fi network. The accessory can initiate this process, but the user must grant permission for the device to join the new Wi-Fi network.

2.1 Requirements

All accessories that support the Wi-Fi Information Sharing feature via iAP2 must send or receive the following iAP2 control session message(s):

- "15.1.1 RequestAccessoryWiFiConfigurationInformation" (page 199)
- "15.1.2 AccessoryWiFiConfigurationInformation" (page 199)

2.2 Usage

To provide Wi-Fi network information, the accessory must support the messages "15.1.1 RequestAccessoryWiFiConfigurationInformation" (page 199) and "15.1.2 AccessoryWiFiConfigurationInformation" (page 199). The device will send "15.1.1 RequestAccessoryWiFiConfigurationInformation" (page 199) to request the Wi-Fi configuration from the accessory. The accessory must respond with a "15.1.2 AccessoryWiFiConfigurationInformation" (page 199) message.

2.3 Test Procedures

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

3 CarPlay

This chapter defines how a vehicle infotainment system (head unit) may receive and interact with digital content (audio, video, and metadata) streamed from a device running iOS. It covers accessories that can receive streamed digital content from 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 integrated with the vehicle audio system and must support audio input and output.

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 Apple CarPlay.

All CarPlay accessories must comply with the *CarPlay Design Guidelines* and *CarPlay Identity Guidelines*.

Aftermarket head units must be integrated systems with a built-in display with a minimum 6 inch diagonal screen size. Aftermarket 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 device for thermal considerations.

3.1 Additional Specifications

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

- USB Specifications
  - USB 2.0 Specification (`https://www.usb.org/document-library/usb-20-specification`)
  - 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
  - "Transmission Control Protocol", RFC 793, September 1981 (`http://www.ietf.org/rfc/rfc793.txt`)
  - "User Datagram Protocol", RFC 768, August 1980 (`http://www.ietf.org/rfc/rfc768.txt`)
  - "Internet Protocol, Version 6 (IPv6)", RFC 2460, December 1998 (`http://www.ietf.org/rfc/rfc2460.txt`)
  - "Definition of the Differentiated Services Field (DS Field) in the IPv4 and IPv6 Headers", RFC 2474, December 1998 (`http://www.ietf.org/rfc/rfc2474.txt`)
  - "The tel URI for Telephone Numbers" RFC 3966, December 2004 (`http://www.ietf.org/rfc/rfc3966.txt`)
  - "Network Time Protocol Version 4: Protocol and Algorithms Specification", RFC 5905, June 2010 (`http://www.ietf.org/rfc/rfc5905.txt`)

  - "Definition of the Opus Audio Codec", RFC 6716, September 2012 (http://www.ietf.org/rfc/rfc6716.txt)
  - "Internet Protocol (IP), DARPA Internet Program, Protocol Specification", RFC 791, September 1981 (http://www.ietf.org/rfc/rfc791.txt)
  - "An Ethernet Address Resolution Protocol (ARP)", RFC 826, November 1982 (http://www.ietf.org/rfc/rfc862.txt)
  - "Unique Local IPv6 Unicast Address", RFC 4193, October 2005 (http://www.ietf.org/rfc/rfc4193.txt)
  - "Dynamic Host Configuration Protocol (DHCP)", RFC 2131, March 1997 (http://www.ietf.org/rfc/rfc2131.txt)
  - "Internet Control Message Protocol (ICMP), DARPA Internet Program, Protocol Specification", RFC 792, September 1981 (http://www.ietf.org/rfc/rfc792.txt)
  - "Internet Control Message Protocol (ICMPv6) for the Internet Protocol Version 6 (IPv6) Specification", RFC 4443, March 2006 (http://www.ietf.org/rfc/rfc4443.txt)
  - "Neighbor Discovery for IP version 6 (IPv6)", RFC 4861, September 2007 (http://www.ietf.org/rfc/rfc4861.txt)
  - "Management Information Base for Network Management of TCP/IP-based internets: MIB-II", RFC 1213, March 1991 (http://www.ietf.org/rfc/rfc1213.txt)
  - "Definition of Managed Objects for the Ether-like Interface Types", RFC 1398, January 1993 (http://www.ietf.org/rfc/rfc1398.txt)
- IEEE 802.11 Standards
  - Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specification, IEEE Std. 802.11-2016
  - Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specification: Enhancements for High Efficiency WLAN, IEEE Std. 802.11ax/D6.1, May 2020
  - Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications Amendment 5: Preassociation Discovery, IEEE Std. 802.11aq-2018, August 2018
- Wi-Fi Alliance Specifications
  - Wi-Fi CERTIFIED™ n Test Plan, version 2.16 or higher (https://www.wi-fi.org/file-member/wi-fi-certified-n-test-plan)
  - Wi-Fi CERTIFIED™ ac Test Plan, version 2.5 or higher (https://www.wi-fi.org/file-member/wi-fi-certified-ac-test-plan)
  - Wi-Fi CERTIFIED™ 6 Test Plan, version 2.5 or higher (https://www.wi-fi.org/file-member/wi-fi-certified-6-test-plan)
  - WPA2 Security Improvements Test Plan, version 1.2 or higher (https://www.wi-fi.org/file-member/wpa2-security-improvements-test-plan)
  - WPA3™ Specification, version 2.0 or higher (https://www.wi-fi.org/file/wpa3-specification)
  - WPA3™ Specification Addendum for WPA3 R3, Draft Version 0.0.3 or higher (https://www.wi-fi.org/file/wpa3-specification-addendum-draft)
  - WPA3™ - SAE Test Plan, version 1.4 or higher (https://www.wi-fi.org/file-member/wpa3-sae-test-plan-v14)

- 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
- ITU Specifications
  - ITU H.264 Specification (http://www.itu.int/rec/T-REC-H.264)
  - ITU-T P.1100 (03/11 or later): Narrow-band hands-free communication in motor vehicles (https://www.itu.int/rec/T-REC-P.1100/en)
  - ITU-T P.1110 (12/09 or later): Wideband hands-free communication in motor vehicles (https://www.itu.int/rec/T-REC-P.1110/en)
  - ITU-T P.1120 (02/17 or later): Super-wideband and fullband hands-free communication in motor vehicle (https://www.itu.int/rec/T-REC-P.1120/en)
- ISO/IEC Standards
  - 14496-3:2009 Information technology – Coding of audio-visual objects – Part 3: Audio, Fourth edition, September 2009 (https://webstore.iec.ch/publication/10018)
  - 15948:2004 Information technology – Computer graphics and image processing – Portable Network Graphics (PNG): Functional specification, First edition, (https://www.iso.org/standard/29581.html)

3.2 General Requirements

3.2.1 Overview

Apple CarPlay lets a device stream an interactive user interface to an accessory with one or more audio/video end-points and controls.

CarPlay is intended for use with displays that the driver can interact with in the front of a vehicle. An accessory must not replicate digital video content to displays elsewhere in the vehicle.

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

3.2.2 Hardware Requirements

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

3.2.2.1 High Resolution Displays

CarPlay provides an interactive User Interface (UI) for a high-resolution main display, and supplementary UI content for a high-resolution instrument cluster display.

Each display must be capable of rendering a UI stream without scaling, support 24 bits of RGB color per pixel, and either a 30 Hz or 60 Hz refresh rate (60 Hz recommended). Displays with square pixels are recommended, but the device can accommodate for non-square pixel displays as well. See "3.2.7.1 User Interface Streams" (page 62).

The accessory must provide a main display which has a minimum resolution of 800 x 480.

3.2.2.2 Processing

The accessory must be capable of:

- Hardware decoding of H.264 video, see "3.2.7.1 User Interface Streams" (page 62).
- Running the Communication Plug-in and other software clients, see "3.2.6 Software Clients" (page 59).
- Providing different audio processing based on the audio mode, see "3.2.7.2 Audio" (page 78).
- Synchronizing the audio output and input sample clocks, see "3.3.2.9 Synchronization" (page 127).

3.2.2.3 User Input Devices

The accessory must provide a touchscreen, a touchpad, or a combination of a rotary knob and at least select and back buttons. See "3.3.5 User Input" (page 136).

The input from all user input devices supported by CarPlay and available in the vehicle must be provided to CarPlay. For example, if a touchscreen, a touchpad with character recognition, a knob, and a back button are available, all of these must be provided to CarPlay.

3.2.2.4 Speakers and Microphone

DEVELOPER PREVIEW

The accessory must provide audio output and input via the vehicle's speakers and a microphone in the vehicle cabin. See "3.2.7.2 Audio" (page 78).

The accessory must include a microphone input stream that is always listening, processed by an echo cancelation and noise reduction (EC/NR) module, stored into a circular buffer to support Siri activation events that require historical user utterances. The accessory must echo cancel any audio playing through the accessory's speakers, e.g. media, radio, alerts, ringtones, turn-by-turn, Siri prompts, etc and must reduce cabin noise such as road or fan noise from the microphone input signal before storing it into the circular buffer. The size and format of the circular buffer are configured by the device using SET_PARAMETER, see Table 3-78 (page 126). For more details on setting up microphone input for Siri, see "3.2.7.2.9 Auxiliary Input Audio - Speech Recognition" (page 88).

If the accessory supports voice activation for another voice assistant, the accessory must also support launching Siri by voice and include a voice detection module to detect any active speech, and when a specific keyword is spoken. See "3.3.7.2 Voice Activation from the Accessory" (page 159).

3.2.2.5 Siri Button

DEVELOPER PREVIEW

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 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 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 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.

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 600 ms. This closely mimics the behavior of the 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 device. The 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.

With instant activation, when Siri is launched by pressing a button, the user can start talking as soon as the Siri button is pressed without waiting for audio prompts or visual feedback. Siri will process audio recorded from the point the user pressed the button to when the user stops talking. See "3.3.7.1 Instant Button Activation from the Accessory" (page 157).

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, 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 a 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 "3.3.8.9 requestSiri" (page 167).

3.2.2.6 CarPlay Entry Point

A CarPlay entry point is a physical button, soft button, menu item, or other control that enables the user to access CarPlay.

- The accessory must provide a CarPlay entry point on its default home screen, the screen that is shown when the user presses the accessory's Home button or performs an equivalent action.
- If the accessory has a function menu that is distinct from the default home screen, the accessory must also provide a CarPlay entry point on the function menu. The function menu is the screen that is shown when the user presses the accessory's Menu button or performs an equivalent action.

- The accessory must make the CarPlay entry point visible, by default, the first time the user presses the accessory's Home button, Menu button, or other such button.
- The accessory cannot require the user to scroll to find the CarPlay entry point.
- If the accessory uses a permanent dock that is always present, the CarPlay entry point must be visible without user interaction when a CarPlay session is activated.
- When the CarPlay session is inactive, the CarPlay entry point must be shown in a dimmed or inactive state, or completely hidden.

3.2.2.6.1 CarPlay Label

The CarPlay entry point must include a CarPlay label, a title that reads *Apple CarPlay*.

- The CarPlay label should match the font, size, style, and placement of other text used in the interface; it should not imitate Apple typography. For example, if all capital letters are used on other items, the CarPlay label can also appear in all capital letters: *APPLE CARPLAY*.
- The CarPlay label should be placed on one line. If space is limited and other titles are stacked, the text of the label can be stacked, with *Apple* on one line and *CarPlay* on the next line.

3.2.2.6.2 CarPlay Logo

The accessory may use the CarPlay logo for the CarPlay entry point in accessory screen interfaces.

- If color graphics are featured in the accessory screen interface, the accessory should use the color version of the CarPlay logo.
- If black-and-white graphics or other monochromatic themes are featured in the accessory screen interface, the accessory should use either the white version or the black version of the CarPlay logo, matching the visual style of the interface.

3.2.2.6.3 CarPlay Icon

The accessory may use the CarPlay icon for the CarPlay entry point in vehicle hardware buttons and ports:

- The CarPlay label, a title reading *Apple CarPlay*, should appear in text below the CarPlay icon.
- If the CarPlay icon is not used, the CarPlay label must appear in the hardware button or port that serves as the CarPlay entry point.

Any use of the Apple CarPlay icon, logo, or trademark within native UI requires Apple review and approval. For detailed information, see *Apple CarPlay Identity Guidelines*.

3.2.2.7 Sensors and Location

The accessory must provide location information to the device as described in "12 Addendum: Location Information" (page 196).

The accessory must derive GPRMC and GPGGA data from a GNSS receiver along with other inertial sensor inputs such as a gyroscope, accelerometer and speed sensor. The data must not be derived from any accessory map data. The GNSS receiver must support GPS and GLONASS in all territories.

Accessories supporting CarPlay over wireless must include GNSS receivers. They are strongly recommended for all CarPlay accessories.

The accessory must provide PASCD.

The accessory may provide a GPRMC `Data status` value of 'X' which is specific for a special case of CarPlay.

3.2.2.8 Connection to the device

The accessory must support one of the following connection configurations:

- CarPlay over wireless only (recommended)
- CarPlay over wireless and CarPlay over USB
- CarPlay over USB only

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

3.2.2.9 Device Mount Location

The device should be mounted in an open holder that is away from direct sunlight or any heat source. Air ventilation is recommended. An enclosed compartment with no air circulation, such as the glove box, should be avoided.

3.2.3 Architecture

DEVELOPER PREVIEW

To support CarPlay over wireless, an accessory must support Bluetooth and Wi-Fi, see "3.2.5 CarPlay over Wireless" (page 32).

To support CarPlay over USB, an accessory must:

- Support USB as a USB device, see *USB: Host Mode* in the *Accessory Interface Specification*.
- Provide a USB-C receptacle, USB-A receptacle, or a Lightning connector, see "3.2.4 CarPlay over USB" (page 25).

The accessory communicates with the 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 device for authentication, identification, and exchange of metadata.

The major building blocks of this architecture are illustrated in Figure 3-1 (page 25), with Apple-provided software components highlighted in orange.

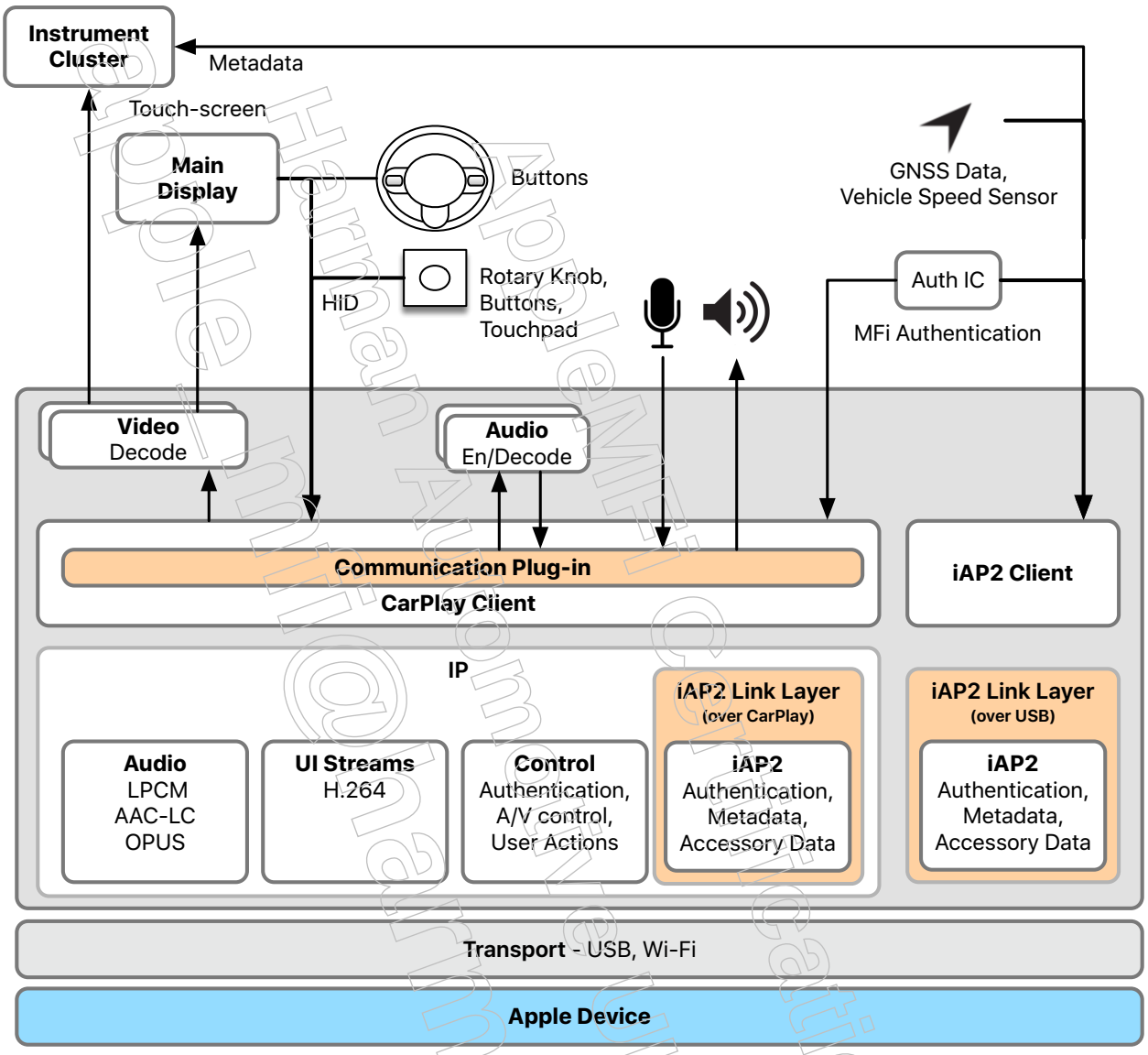

Figure 3-1: Topology of an automotive head unit using CarPlay

Figure 3-1 (page 25) shows an automotive head unit and its associated instrument cluster and displays with:

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

3.2.4 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 device and accessory is recommended.

The accessory must also support the USB Host Mode transport (where the accessory is a USB device and the device is the host), see *USB: Host Mode* in the *Accessory Interface Specification*.

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 device.

For an accessory that normally acts as a USB host, the steps shown in Figure 3-2 (page 27) are required.

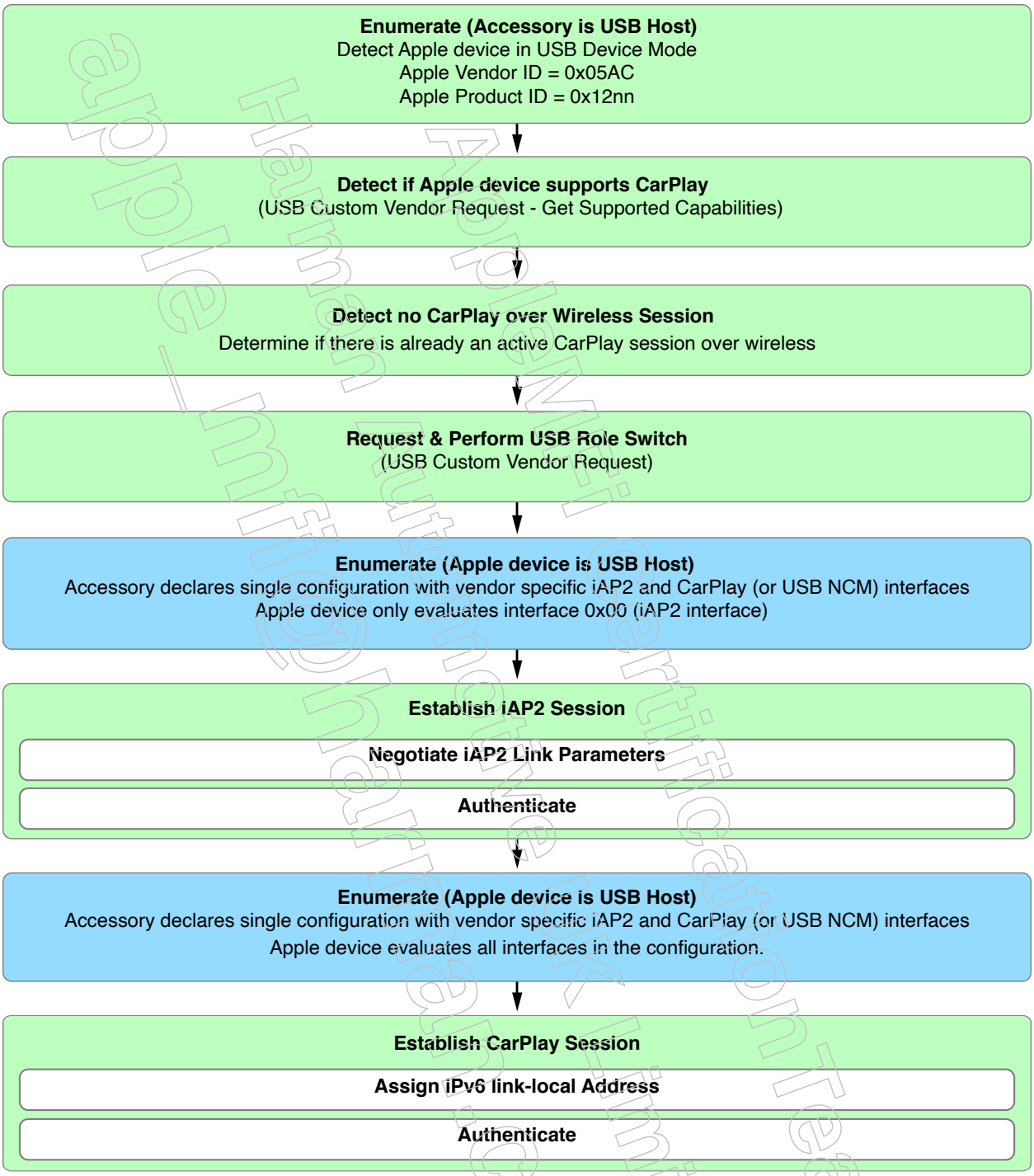

Figure 3-2: Process for establishing a CarPlay session

3.2.4.1 Role Switch

If the accessory normally acts as a USB host, it must perform a host-to-device role switch, see *USB: Role Switch* in the *Accessory Interface Specification*.

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 A variant 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.

3.2.4.2 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* in the *Accessory Interface Specification*).
- A USB NCM control interface (see Table 3-1 (page 28) and Table 3-2 (page 29)).
- A USB NCM data interface (see Table 3-3 (page 29)).

Accessories must not use the NCM interface for anything other than CarPlay.

The accessory must proceed to establish an authenticated iAP2 Session as described in *Accessory Authentication* in the *Accessory Interface Specification*. CarPlay accessories must advertise support for control session version 2 (see *iAP2: Control Session* in the *Accessory Interface Specification*).

Note: When the device is using control session version 2, the device may access other USB interfaces before authentication is complete. When using control session version 1, the device will wait until authentication is completed successfully before accessing other USB interfaces.

Table 3-1: USB NCM Control Interface Descriptor

USB Descriptor	Requirement	Description
Interface Number	0xNN	Must be different from the iAP2 interface and USB NCM data interface numbers. Must match the USBHostTransportCarPlayInterfaceNumber, see *Accessory Identification* in the *Accessory Interface Specification*
Interface Class	0x02	USB Communication Interface Class
Interface Subclass	0x0D	Network Control Model
Interface Protocol	0x00	No encapsulated commands / responses
Number of Endpoints	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 3-2 (page 29).

Table 3-2: 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
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 device to use via the CDC Ethernet Networking functional descriptor. The accessory must not use the same MAC address provided to the 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 device and the head unit. MAC addresses must be unique between individual accessories. An accessory must use the same MAC address for every connection.

Table 3-3: 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 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 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 (iAP 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
```

3.2.4.3 Authentication

CarPlay is an authenticated solution, requiring the use of an Apple Authentication Coprocessor obtained through Apple. Devices will stream only to authorized accessories. CarPlay accessories using Apple Authentication Coprocessor 3.0 may only claim compatibility with iOS 10.3 or later.

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 a device will be correctly decoded and displayed and that all electrical requirements described in this specification are met.

3.2.4.4 Networking

To establish a CarPlay session, the accessory must be networked with the 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.

3.2.4.5 Session Establishment

After the accessory has detected that CarPlay is available on the connected device, it must request a session initiation and provide the required parameters in the "15.6.2 CarPlayStartSession" (page 217) message.

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* in the *Accessory Interface Specification*.

For more information on setting up a CarPlay session, see "3.3.2 Setup and Control" (page 95).

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

3.2.4.6 Session Termination

The accessory must be able to detect a disconnect and terminate the session within 500 ms of a physical detachment of the device.

The accessory must not change the audio state as part of terminating a session.

3.2.5 CarPlay over Wireless

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

Setup for CarPlay over wireless begins with Bluetooth discovery between the 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 device the user initiates pairing via the iOS Settings app. The accessory and the 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 device and the accessory.

In addition, the pairing process can be simplified by just plugging in the device over USB and executing Out-Of-Band Bluetooth Pairing as described in *Accessory Interface Specification*.

After Bluetooth pairing is completed, the accessory connects iAP2 over Bluetooth and declares support for CarPlay over wireless. Once the device confirms that CarPlay over wireless is available, the accessory request a session initiation and provides the Wi-Fi credentials and the IP address to be used for the connection.

The device will join the accessory's access point and establish a CarPlay session.

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 device to associate with the already known Wi-Fi network. Once the accessory confirms that wireless CarPlay is enabled on the device, it initiates the session in the same way as during initial pairing.

For accessories that support multiple paired devices, including Bluetooth devices, devices supporting CarPlay over USB and CarPlay over wireless, the user may select which device to use for CarPlay from a list presented in the accessory's user interface. After a successful pairing, the accessory must add the CarPlay device to the lists as the last used device on the system. Upon reconnection the accessory must reconnect to the last used device. If a device is connected over USB and over wireless, it appears only once in the list. For more details see the CarPlay Design Guidelines.

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, see "3.2.6.2 iAP2 Client over Bluetooth" (page 60).

After starting iAP2, accessories using 5 GHz Wi-Fi access points for CarPlay may negotiate all of the relevant Bluetooth profiles such as HFP, A2DP, AVRCP, etc. Accessories using 2.4 GHz must only negotiate the iAP2 profile and no other legacy profiles. However, once a CarPlay session is established, the device will notify the accessory to disconnect all active profiles, see "3.3.8.3 disableBluetooth" (page 163).

During reconnections for CarPlay, the accessory must start iAP2 prior to any additional Bluetooth profiles.

Accessories must use "15.6.1 CarPlayAvailability" (page 216) to receive notifications about the availability of CarPlay over wireless on a 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 device.

Accessories implementing CarPlay over wireless and CarPlay over USB may use "15.6.1 CarPlayAvailability" (page 216) to keep a list of devices that were previously paired over either wireless or USB transport.

See *Bluetooth* in the *Accessory Interface Specification* for additional requirements.

3.2.5.0.1 Accessory CarPlay Bluetooth EIR

The 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 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 "3.1 Additional Specifications" (page 18).

3.2.5.0.2 Device CarPlay Bluetooth EIR

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.

3.2.5.0.3 Accessory iAP2 Bluetooth EIR

The accessory must support iAP2 over Bluetooth, see "3.2.6.2 iAP2 Client over Bluetooth" (page 60).

3.2.5.1 Wi-Fi Access Point

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

During initial pairing, the accessory provides the Wi-Fi access point's Wi-Fi credentials (SSID and Passphrase) using iAP2 over Bluetooth, see "3.2.6.2 iAP2 Client over Bluetooth" (page 60). These Wi-Fi credentials are stored on the device and will be used to join the Wi-Fi access point automatically during reconnection. To speed up the reconnection scenario, the device will initiate Wi-Fi scanning based on a re-connect of any Bluetooth profile.

Once the device joins the Wi-Fi access point, it waits for the accessory to request a session initiation and provide the required parameters in the "15.6.2 CarPlayStartSession" (page 217) message.

The accessory must ensure that the Wi-Fi access point is fully operational before it sends the "15.6.2 CarPlayStartSession" (page 217) message to the device to request a session initiation.

During reconnection, the accessory must ensure that the Wi-Fi access point is fully operational before Bluetooth reconnection, as the device will initiate Wi-Fi association as soon as any Bluetooth profile is connected with the accessory.

The Wi-Fi access point must support the following based on the number displays supported by the system (as measured with iperf):

- Main display only: 25 Mbps bandwidth 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.
- Main display and one UI stream on an instrument cluster display: 35 Mbps bandwidth 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 35 Mbps UDP uplink stream.
- Main display and two UI streams on an instrument cluster display: 45 Mbps bandwidth 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 45 Mbps UDP uplink stream.

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 be configured with a Beacon Interval of 100ms.

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

3.2.5.1.1 Hardware Requirements

It is recommended that the accessory supports Wi-Fi 6 (IEEE 802.11ax) 5 GHz HE80 or HE40 as defined in the *IEEE 802.11ax/D6.1* standard.

It is recommended that the accessory uses a MIMO (2x2) hardware configuration.

The accessory may support Wi-Fi 5 (IEEE 802.11ac) 5 GHz or Wi-Fi 4 (IEEE 802.11n) 5 GHz as defined in the *IEEE 802.11-2016* standard. If the accessory supports these Wi-Fi standards it is recommended to support 80MHz or 40MHz channel width protocols:

- Wi-Fi 5 (IEEE 802.11ac) 5 GHz VHT80 or VHT40
- Wi-Fi 4 (IEEE 802.11n) 5 GHz HT40

The accessory may support 20MHz channel width protocols, described below for each Wi-Fi standard:

- Wi-Fi 6 (IEEE 802.11ax) 5 GHz HE20
- Wi-Fi 5 (IEEE 802.11ac) 5 GHz VHT20
- Wi-Fi 4 (IEEE 802.11n) 5 GHz HT20

The accessory must at a minimum support Wi-Fi 4 (IEEE 802.11n) 5 GHz HT20.

Accessories must not use configurations which support only Wi-Fi 4 (IEEE 802.11n) 2.4 GHz HT20 or Wi-Fi 6 (IEEE 802.11ax) 2.4 GHz HE20, unless the accessory is operating in a region where regulatory restrictions prevent the use of Wi-Fi in the 5 GHz frequency band in vehicular environments.

Accessories must not support Wi-Fi 4 (IEEE 802.11n) 2.4 GHz HT40.

For more information on the *IEEE 802.11-2016* and *IEEE 802.11ax/D6.1* standards, see "3.1 Additional Specifications" (page 18).

3.2.5.1.2 Frequency Bands

Accessories operating in regulatory domains mandating 2.4 GHz, must operate the Wi-Fi access point in one of the channels listed in Table 3-4 (page 35).

Table 3-4: 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 3-5 (page 35).

Table 3-5: 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
UNII-III (Upper Band)	157	5.785 GHz
UNII-III (Upper Band)	161	5.805 GHz
UNII-III (Upper Band)	165	5.825 GHz

Selecting a new operating channel must be avoided across accessory power cycles as this increases reconnection time. Switching the operating channel during an active CarPlay session may be beneficial for specific use cases, see "3.2.5.2 Advanced Wi-Fi Protocol Features" (page 46).

3.2.5.1.3 Multiple Access Points

If the accessory operates in both the 2.4 GHz and the 5 GHz frequency bands simultaneously and is configured with the same SSID, security mode, and password, then both frequency bands must support CarPlay over wireless. In this configuration the accessory must always request a session initiation using a channel in the 5 GHz band (see Table 15-38 `CarPlayStartSessionWirelessAttributes` parameter group (page 218)) and it is recommended that both frequency bands provide the same network services (e.g. Internet data access).

If the accessory operates in both the 2.4 GHz and the 5 GHz frequency bands simultaneously and is not configured with the same SSID, then CarPlay over wireless must only be supported on the 5 GHz frequency band.

3.2.5.1.4 Wi-Fi Requirements

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

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

Quality of Service Requirements

The accessory must support the WFA Wireless Multimedia (WMM) Quality of Service (QOS) mechanism, see "3.1 Additional Specifications" (page 18). The CarPlay protocol uses AC_VO for Voice data traffic, AC_VI for Screen data traffic, and AC_VO for Control data traffic.

Power Saving Requirements

The accessory must support standard power management and power save functions as defined in the *IEEE 802.11-2016* standard, see "3.1 Additional Specifications" (page 18).

When the device transmits a null data packet with the PM Bit set (entering IEEE 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 device.

3.2.5.1.5 Security

Wi-Fi Protected Access (WPA) Modes

DEVELOPER PREVIEW

The accessory's access point(s) must support both WPA3 Personal Transition security mode and WPA3 Personal only security mode as defined in *Wi-Fi Alliance WPA3™ Specification*. The accessory must use WPA3 Personal Transition security mode as the default security mode for CarPlay. If the accessory allows the user to choose a security mode, WPA3 Personal Only security mode must be user configurable.

When initiating a CarPlay session the accessory must report which security mode is in use through the "15.6.2 CarPlayStartSession" (page 217) message.

Protected Management Frames

The CarPlay access point(s) must support IEEE 802.11 Protected Management Frames (PMF) as defined in *IEEE 802.11-2016*. PMF must be supported and enabled for all WPA2 and WPA3 security modes.

Security Related Tallies and Counters

The CarPlay access point(s) may support the mandatory security related tallies/counters defined in the IEEE 802.11 Management Information Base (MIB) as defined in *IEEE 802.11-2016*.

Hardware Based Encryption

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

Government Standard Encryption

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

Government Recommendations for Key Management

The implementation may follow government guidelines for use and implementation of cryptographic mechanisms, as per NIST Special Publication 800-57 Part 1 Revision 5.

Secure Random Number Generation

The CarPlay access point(s) must use a secure random number generator to generate all random numbers used for authentication, encryption, or privacy purposes. This induces all authentication and encryption keys, and values used to derive such keys, such as nonces. These constructs must comply with the latest versions of NIST SP 800-90A, NIST SP 800-90B, and NIST SP 800-90C, as appropriate.

Security Modes

The following security modes must not be used for CarPlay over wireless operation:

- None/Open
- WEP Security mode
- WPA Personal only
- WPA/WPA2 Personal mixed
- WPA2 Personal only mode
- No Enterprise based security modes
  - WPA Enterprise
  - WPA/WPA2 Enterprise mixed
  - WPA2 Enterprise only
  - WPA3 Enterprise only
  - WPA3 Enterprise Transition
  - WPA3 Enterprise 192-bit
- WEP40 or WEP104 Cipher Suites
- Temporal Key Integrity Protocol (TKIP) Cipher Suite

3.2.5.1.6 Privacy

Protecting CarPlay Clients

MAC Address Management

In order to protect the privacy of the CarPlay client, the accessory must:

- Support clients associating with randomized MAC addresses, which may be rotated between each association to the CarPlay access point.
- Not filter or restrict access to the CarPlay access point based on the device's MAC address.

- Not identify the device by its MAC address, or conditionalize features or behavior based on the device's MAC address.
- Not remember the MAC address of unassociated clients once their DHCP lease times out.

Wi-Fi Client Privacy

The CarPlay accessory implementation must support clients associating with client privacy protections as defined by *Wi-Fi Alliance WPA3™ Specification Addendum for WPA3 R3* and *IEEE Std. 802.11aq-2018*.

Protecting CarPlay Access Points

MAC Address Randomization

The CarPlay accessory may randomize the MAC address of the CarPlay access point(s).

If implemented, then randomization must:

- Occur at each boot, before the access point(s) start beaconing
- Set all bits of the MAC address to a random value from a secure random number generator, except for:
  - The Locally Administered bit, which shall be set to "1", and
  - The Unicast/Multicast bit, which shall be set to "0"
- Set each CarPlay access point(s) MAC address to a unique, random value that is not correlated with any of the other CarPlay access point(s) that may be present in the vehicle.

Information Elements

A CarPlay access point must list all used Information Elements in ascending order by element ID and OUI.

A CarPlay access point must not use the following fields in the Apple Device Information Element:

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

It is not recommended that a CarPlay access point use the Interworking Information Element for privacy reasons.

IEEE 802.11 Time Synchronization Function (TSF)

A CarPlay access point should randomize the initial value of the IEEE 802.11 Time Synchronization Function (TSF) to prevent observers from using the TSF clock to determine how long the vehicle has been in operation, or otherwise fingerprint and track the vehicle and user behavior.

If implemented, then randomization must:

- Occur at each boot, before the access point(s) start beaconing
- Initialize the lower 56 bits of the IEEE 802.11 TSF to a random value from a secure random number generator.
- Set each CarPlay access point(s) initial TSF to a unique, random value that is not correlated with any of the other CarPlay access point(s) that may be present in the vehicle.

IEEE 802.11 Sequence Numbers

A CarPlay access point may randomize their initial IEEE 802.11 sequence number to minimize the ability to identify an access point.

If implemented, then randomization must:

- Occur at each boot, before the access point(s) start beaconing
- Randomly initialize all bits of the IEEE 802.11 sequence number to a random value from a secure random number generator.
- Set each CarPlay access point(s) initial sequence number to a unique, random value that is not correlated with any of the other CarPlay access point(s) that may be present in the vehicle.

IEEE 802.11 Dialogue Tokens

A CarPlay access point should randomize the initial IEEE 802.11 dialogue token used in any protocol exchanges initiated by the access point to minimize the ability to identify the access point or how long it has been in operation.

If implemented, then randomization must:

- Occur at each boot, before the access point(s) start beaconing
- Randomly initialize all bits of the first IEEE 802.11 dialogue token to a random value from a secure random number generator
- For each subsequent exchange, generate a new random dialogue token.
- Set each CarPlay access point(s) initial dialogue token to a unique, random value that is not correlated with any of the other CarPlay access point(s) that may be present in the vehicle.

IEEE 802.11 Scrambler Seed

A CarPlay access point may randomize the IEEE 802.11 scrambler seed to minimize the ability to identify the access point.

If implemented, then the implementation must use a secure random number generator to generate a new, random scrambler seed for each transmitted IEEE 802.11 packet.

Wi-Fi Client Privacy

If the CarPlay access point(s) can issue Wi-Fi scans, it may implement Wi-Fi Client Privacy mechanisms defined by *Wi-Fi Alliance WPA3™ Specification Addendum for WPA3 R3* and *IEEE Std. 802.11aq-2018* when issuing such scans.

Scan Timing and Ordering

If the CarPlay access point can issue Wi-Fi scans, it should randomize the time interval between scans and the order in which channels are scanned to protect user privacy.

Identifiers and Fingerprinting

To prevent fingerprinting, CarPlay access point may not include unique identifiers (e.g serial numbers), identifying information (e.g. firmware versions, car make or model), or uncommon attributes (e.g. non-required vendor-specific capabilities or IEs) in frames that are not protected by IEEE 802.11 Management Frame Protection or the secure data path.

This includes information contained in the 802.11 Information Elements carried in frames such as Beacons, Probe Responses, Authentication, Association, or the 4-way handshake's EAPOL-Key frames. Such information should be exchanged post-association, using secure data packets (preferred) or IEEE 802.11 protected management frames.

Protecting Accessories when Operating as a Standard Wi-Fi Client

Wi-Fi Client Privacy

If the accessory can act as a Wi-Fi client, it may implement the Wi-Fi Client Privacy mechanisms as defined by *Wi-Fi Alliance WPA3™ Specification Addendum for WPA3 R3* and *IEEE Std. 802.11aq-2018* when operating as a client or issuing client scans.

Scans

If the accessory can issue Wi-Fi client scans, it may randomize the time interval between scans and the order in which channels are scanned to protect user privacy.

Identifiers and Fingerprinting

To prevent fingerprinting, the Wi-Fi client may not include unique identifiers (e.g serial numbers), identifying information (e.g. firmware versions, car make or model) or uncommon attributes (e.g. non-required vendor-specific capabilities or IEs) in frames that are not protected by IEEE 802.11 Management Frame Protection or the secure data path.

This includes information contained in the IEEE 802.11 Information Elements carried in frames such as Probe Requests, GAS requests, ANQP requests, Authentication, Association, or the 4-way handshake's EAPOL-Key frames. Such information should be exchanged post-association, using secure data packets (preferred) or IEEE 802.11 protected management frames.

3.2.5.1.7 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 3-6: Expected Throughput for IEEE 802.11 MAC/PHY Modes

IEEE 802.11 MAC/PHY Radio Modes	(1×1) 1SS		(2×2) 2SS	
	TCP	UDP	TCP	UDP
Wi-Fi 4 (IEEE 802.11n) 2.4 GHz and 5 GHz - HT20	50	60	105	120
Wi-Fi 4 (IEEE 802.11n) 5 GHz - HT40	110	125	220	250
Wi-Fi 5 (IEEE 802.11ac) 5 GHz - VHT20	60	70	125	140
Wi-Fi 5 (IEEE 802.11ac) 5 GHz - VHT40	140	160	280	320
Wi-Fi 5 (IEEE 802.11ac) 5 GHz - VHT80	320	360	600	720
Wi-Fi 6 (IEEE 802.11ax) 2.4 GHz and 5 GHz - HE20	105	120	215	240
Wi-Fi 6 (IEEE 802.11ax) 5 GHz - HE40	215	240	430	485
Wi-Fi 6 (IEEE 802.11ax) 5 GHz - HE80	440	500	480	950

3.2.5.1.8 Wi-Fi Alliance Compliance and Conformance

DEVELOPER PREVIEW

A CarPlay accessory Wi-Fi implementation supporting Wi-Fi 6 (IEEE 802.11ax) must be in compliance with the mandatory to implement requirements defined in the latest versions of *Wi-Fi CERTIFIED™ 6 Test Plan*.

A CarPlay accessory Wi-Fi implementation supporting Wi-Fi 5 (IEEE 802.11ac) must be in compliance with the mandatory to implement requirements defined in the latest versions of *Wi-Fi CERTIFIED™ ac Test Plan*.

A CarPlay accessory Wi-Fi implementation supporting Wi-Fi 4 (IEEE 802.11n) must be in compliance with the mandatory to implement requirements defined in the latest versions of *Wi-Fi CERTIFIED™ n Test Plan*.

A CarPlay accessory Wi-Fi implementation supporting WPA3 Personal Transition security mode must be in compliance with the mandatory to implement requirements defined in the latest version of *Wi-Fi WPA3-SAE Test plan version 1.3*.

A CarPlay accessory Wi-Fi implementation supporting WPA2 Personal security mode must be in compliance with the mandatory to implement requirements defined in the latest version of *WPA2 Security Improvements Test Plan.*.

If the accessory's access point uses SISO (1x1) configuration, then it must be certified as a Mobile Access Point for the appropriate *Wi-Fi CERTIFIED™ ac Test Plan* or *Wi-Fi CERTIFIED™ n Test Plan*.

If the accessory's access point uses at least a MIMO (2x2) configuration:

- Wi-Fi 6 (IEEE 802.11ax) implementations must be certified as a Mobile Access Point using the *Wi-Fi CERTIFIED™ 6 Test Plan*.
- Wi-Fi 5 (IEEE 802.11ac) and Wi-Fi 4 (IEEE 802.11n) implementations must be certified as a Standard Access Point for the appropriate *Wi-Fi CERTIFIED™ ac Test Plan* or *Wi-Fi CERTIFIED™ n Test Plan*.

3.2.5.1.9 Interworking Information Element (IE)

Deprecated: The use of Interworking IE is deprecated, see "3.2.5.1.6 Privacy" (page 38).

The accessory must set the following fields, see *IEEE 802.11-2016*:

- "Element ID" must be set to 107.
- "Length" must be set to 3.
- "Access Network Options" field:
  - "Network Access Type" and "Internet" must be set based on the availability of Internet connectivity. For more information, see "3.2.5.4 Enabling Internet Data Connectivity" (page 50).
  - "ASRA" must be set to 0.
  - "ESR" must be set to 0.
  - "UESA" must be set to 0.
- "Venue Info" is a 2-octet field which must include:
  - "Venue Group" must be set to 10 (Vehicular).
  - "Venue Type" must be set as specified in Table 3-7 (page 43).

Table 3-7: Venue Type

Value	Description
1	Automobile or Truck
3	Bus
7	Motor Bike

For more information on the Interworking IE, see "3.1 Additional Specifications" (page 18).

3.2.5.1.10 Apple Device Information Element (IE)

The accessory must support and include a vendor-specific IEEE 802.11 IE using the OUI 00-A0-40 (registered to Apple Inc.). The payload portion of the IE is composed of sub-IEs defined in this section.

The accessory must include the Device IE in the following IEEE 802.11 management frames at all times during the operation of the Wi-Fi access point:

- Beacon
- Probe response
- Association response
- Re-Association response

The following parameters are required:

- Deprecated: Name

- Deprecated: Manufacturer
- Deprecated: Model
- Deprecated: OUI
- Deprecated: Bluetooth MAC Address
- Deprecated: Device ID
- Features flags as specified in Table 3-8 (page 44)

Table 3-8: Apple Device IE overall structure

Name	Size	Value	Description
Element ID	1	0xDD	Vendor-specific element ID as specified in *Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specification, IEEE Std. 802.11 - 2007.*
Length	1	Variable	Number of bytes in IE (excludes element ID and length bytes).
OUI	3	0x00 0xA0 0x40	Apple Inc. OUI reserved for this IE.
Sub-type	1	0x00	Sub-type of the 00-A0-40 Apple Inc. OUI.
Elements:	Variable	Variable	Sub IE elements defined by this specification.

Table 3-9: Apple Device IE element structure

Name	Size	Description
Element ID	1	Vendor specific element ID as specified in *Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specification, IEEE Std. 802.11 - 2007.*
Length	1	Number of bytes in the element payload (excludes element ID and length bytes).
Payload	Variable	Payload defined by the element ID.

Table 3-10: Apple Device IE elements

Element ID	Name	Format	Description
0x00	Flags	n:bits	Flags about the accessory: b0-b7, b8-b15, etc. See Table 3-11 (page 46). Each flag is a bit. Bit numbering starts from the leftmost bit of the first byte and uses the minimum number of bytes needed to encode the bits. For example: If only bit 1 is set, it would be 0x40. If bit 1 (0x40) and bit 7 (0x01) are set, it would be 0x41. If bit 1 (0x40), bit 7 (0x01), and bit 10 (0x0020) are set, it would be 0x41, 0x20. If only bit 10 (0x0020), it would be 0x00, 0x20.

0x01	Name	UTF-8	Deprecated: Friendly name of the accessory. This should only be provided if the user configured a custom name or the firmware of the accessory has reason to believe it can provide a name that is better than the default name the client software will provide for it based on the model. Due to localization issues, it often better to only provide this element if the user has configured a name.
0x02	Manufacturer	UTF-8	Deprecated: Machine-parsable manufacturer of the accessory (e.g. "Manufacturer").
0x03	Model	UTF-8	Deprecated: Machine-parsable model of the accessory (e.g. "AB1234").
0x04	OUI	3 bytes	Deprecated: OUI of the accessory including this IE. (e.g. 0x00 0xA0 0x40)
0x05			Reserved
0x06	Bluetooth MAC	6 bytes	Deprecated: MAC address of the Bluetooth radio, if applicable.
0x07	Device ID	6 bytes	Deprecated: Globally unique ID for the accessory (e.g. the primary MAC address, such as 00:11:22:33:44:55). This should be the primary MAC address of the device. If the device has multiple MAC addresses, one must be chosen as the primary MAC address such that it never changes (e.g. does not depend on the network interface currently active).
0x08-0xFF			Reserved.
0xDD	Vendor-specific	n bytes	Deprecated: Same format as a normal vendor-specific IE element.

Table 3-11: Flags

Value	Bit	Description
0x80	0	Reserved.
0x40	1	Reserved.
0x20	2	Reserved.
0x10	3	Reserved.
0x08	4	Reserved.
0x04	5	Reserved.
0x02	6	Reserved.
0x01	7	Reserved.
0x0080	8	Reserved.
0x0040	9	Reserved.
0x0020	10	Supports CarPlay over Wireless.
0x0010	11	Reserved.
0x0008	12	Reserved.
0x0004	13	Reserved.
0x0002	14	Supports 2.4 GHz Wi-Fi networks.
0x0001	15	Supports 5 GHz Wi-Fi networks.
0x000080	16	Reserved.
0x000040	17	Reserved.
0x000008	20	Reserved.
0x000004	21	Reserved.
0x000002	22	Reserved.
0x000001	23	Reserved.

3.2.5.2 Advanced Wi-Fi Protocol Features

3.2.5.2.1 Use Cases

Accessories supporting CarPlay over wireless may encounter situations where they may need to utilize advanced Wi-Fi protocol features.

Interference Detection and Mitigation

Wireless interference may impact the wireless CarPlay performance and user experience. Wireless interference may originate from inside or outside the vehicle. Wireless interference may be coherent or incoherent and may occur dynamically and for an unknown duration of time. The accessory may be able to detect such interference and switch the CarPlay access point to a new operating Wi-Fi channel.

Regulatory Domain Change during CarPlay Session

In some situations, a regulatory domain change of the Wi-Fi operating channel might be required during an active CarPlay over wireless session. The accessory may detect the need for change the operating Wi-Fi channel and either switch the CarPlay access point to a new channel or switch CarPlay operation to a new access point.

Load Balancing and Client Steering

Accessories deploying more than one Wi-Fi access point, may need to load balance Wi-Fi clients between the different access points. In addition, if a wireless CarPlay client connects to an access point operating in the 2.4 GHz frequency band, the accessory may steer the client to the access point operating in the 5 GHz frequency band in order to avoid interference and due to load balancing purposes.

In any of the above situations, the accessory may choose to use some of the advanced Wi-Fi protocol features to update the operating CarPlay Wi-Fi channel:

- If the accessory operates a single Wi-Fi access point, it must use Channel Switch Announcement (CSA) and Extended CSA (E-CSA) to switch the operating channel (see "3.2.5.2.2 Channel Switch Announcement and Extended Channel Switch Announcement" (page 47)). CSA and E-CSA cannot be used for load balancing and client steering.
- If the accessory operates multiple Wi-Fi access points, it must use BSS Transition Management (BTM) to switch the operating channel (see "3.2.5.2.5 BSS Transition Management" (page 48)). If BTM is not supported, the accessory may implement Neighbor Request and Report (NR) (see "3.2.5.2.4 Neighbor Request and Report" (page 48)). The accessory may implement Beacon Request and Report (BR) (see "3.2.5.2.3 Beacon Request and Report" (page 48)) in conjunction with BTM and NR functionality.

3.2.5.2.2 Channel Switch Announcement and Extended Channel Switch Announcement

A CarPlay accessory may support (as defined in *IEEE 802.11-2016*):

- Channel Switch Announcement element
- Channel Switch Announcement frame (Action frame) format
- Extended Channel Switch Announcement element
- Extended Channel Switch Announcement frame (Action frame) format

3.2.5.2.3 Beacon Request and Report

A CarPlay accessory may support (as defined in *IEEE 802.11-2016*):

- Beacon Request and Report protocol
- Beacon Request and Report frame

3.2.5.2.4 Neighbor Request and Report

The Neighbor Request and Report protocol may be used in a system with multiple CarPlay access points.

A CarPlay accessory may support (as defined in *IEEE 802.11-2016*):

- Neighbor Request and Report protocol
- Neighbor Request feature
- Neighbor Report feature

3.2.5.2.5 BSS Transition Management

The BSS Transition Management protocol may be used in a system with multiple CarPlay access points. The BTM exchange may be triggered by the wireless CarPlay client, STA, using a BTM Query message to the wireless CarPlay access point. The BTM exchange may also be triggered by the wireless CarPlay access point by sending an unsolicited BTM Request message to the wireless CarPlay client.

A CarPlay accessory may support (as defined in *IEEE 802.11-2016*):

- BSS Transition Management protocol
- BSS Transition Management Query frame format
- BSS Transition Management Request frame format

A wireless CarPlay client may send a BTM Query frame to the wireless CarPlay access point to request a prioritized list of wireless CarPlay BSS's within the access point ESS to which the wireless CarPlay client may transition. After receiving a BTM Query frame, the wireless CarPlay access point shall respond with a BTM Request frame as defined in *IEEE 802.11-2016*. The BTM Request frame shall include a BSS Transition Candidate List Entry field, that contains one or more Neighbor Report elements.

A wireless CarPlay access point may send an unsolicited BTM Request frame to an associated wireless CarPlay client when it would like to steer the client to a new BSS and before terminating the BSS. The unsolicited BTM Request frame shall include the recommended transition channels and BSS's using the BSS Transition Candidate List containing zero or more Neighbor Report elements.

A wireless CarPlay access point, prior to disassociating a wireless CarPlay client, shall send an unsolicited BTM Request frame with the Disassociation Imminent field set to one to the wireless CarPlay client. When the Disassociation Imminent field is set to one, the wireless CarPlay access point shall set the Disassociation Timer field to the number of TBTTs that will occur prior to the wireless CarPlay access point disassociating the wireless CarPlay client.

The wireless CarPlay access point, prior to terminating the BSS, shall send an unsolicited BTM Request frame to all associated wireless clients by setting the BSS Termination Included field to one. When the BSS Termination Included field is set to one, the wireless CarPlay access point shall set the BSS Termination TSF field of the BSS Termination Duration field according to the time until the BSS is terminated and terminate the BSS after the BSS Termination TSF is reached.

When a wireless CarPlay client receives an unsolicited BTM Request frame, it may accept or reject the transition management request frame by sending a BTM Response frame containing the Status Code field indicating acceptance or rejection. If the wireless CarPlay client accepts the BTM Request frame, it shall attempt to associate with the indicated BSS's before the Disassociation Timer expires or the BSS Termination TSF is reached.

A wireless CarPlay client may include in the BSS Transition Response frame a BSS Transition Candidate List containing Neighbor Report elements to inform the wireless CarPlay access point of BSS's the wireless client has discovered in its scanning process.

On receiving BTM request, CarPlay client initiates scans and chooses a candidate if the potential candidate is better than the connected CarPlay access point in terms of signal strength, load and other static parameters. CarPlay client always prefers 5Gh BSS's if they are in proximity.

Throttling Mechanism

To avoid frequent roams, CarPlay clients implement a throttling mechanism where-in it does not honor every BTM request sent by the CarPlay access point. When it rejects the BTM request, it sends appropriate reason code.

Preferred Candidate List

CarPlay clients read the preferred candidate list and its preference value on the received BTM request. If the preference value of a preferred candidate is 0, that candidate is pruned from potential roam candidate list.

Abridged Bit

If the Abridge bit is set in a BTM request, CarPlay clients select the channels from the preferred candidate list and scans only those channels. CarPlay clients may join a BSS which may not be present in the preferred candidate list provided by the CarPlay access point. CarPlay clients send the newly chosen BSS in its BTM response.

BTM Rejection

CarPlay clients can reject BTM requests when it is performing critical tasks, to avoid frequent roams of when it is not able to find any suitable roam candidate. CarPlay access points are recommended to make use of the reason codes before retrying the BTM requests:

- To avoid frequent roams, CarPlay clients implement a throttling mechanism where by it does not honor every BTM request sent by the CarPlay access point. A reject reason 1 will be included in the BTM response frame.
- On receiving BTM request, CarPlay clients may not find any suitable candidate in its evaluation. It sends BTM reject response with reason code 6. CarPlay clients attach the list of candidates it found to the BTM response. CarPlay access points can make use of this list to influence decisions such as when to retry BTM requests.
- On receiving BTM request, if CarPlay clients do not find any candidate in its evaluation, it sends a BTM reject response with reason code 7. No candidate list will be attached to BTM reject response.

Table 3-12 (page 50) defines the BTM status codes.

Table 3-12: BTM status codes

Status code	Description
0	Accept
1	Reject: Unspecified reason
2	Reject: Insufficient Beacon or Probe Response frames received from all candidates
3	Reject: Insufficient available capacity from all candidates
4	Reject: BSS Termination undesired
5	Reject: BSS Termination delay requested
6	Reject: STA BSS Termination Candidate list provided
7	Reject: No suitable BSS transition candidates
8	Reject: Leaving ESS
9-255	Reserved

3.2.5.3 Networking

To establish a CarPlay session, the accessory must be networked with the 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 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)*.

Accessories must continue to support IPv4 DHCP addressing for backwards compatibility with older iOS devices. The accessory must store 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.

The CarPlay accessory must handle exhaustion of the DHCP address pool by reclaiming unused addresses from unassociated clients.

If the accessory provides internet connectivity it must provide routable IP addresses.

3.2.5.4 Enabling Internet Data Connectivity

Deprecated: Accessories are no longer required to indicate if they are providing an active internet connection through Interworking Information Element (IE) fields.

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.

The accessory must set the values of "Network Access Type" and "Internet" fields in the Interworking IE based on the availability of internet connectivity (see "3.2.5.1.9 Interworking Information Element (IE)" (page 42)):

- Accessory never offers internet connectivity
  - "Network Access Type" set to 4 (Personal device network)
  - "Internet" set to 0
- Accessory offers internet connectivity and the provided data plan does not expire throughout the lifetime of the vehicle
  - "Network Access Type" set to 3 (Free public network)
  - "Internet" set to 1
- Accessory offers internet connectivity and the provided data plan can expire
  - "Network Access Type" set to 2 (Chargeable public network)
  - "Internet"
    * If the data plan is active: "Internet" set to 1
    * If the data plan has not been activated, has expired or is not valid: "Internet" set to 0

Note: Do not toggle the value of "Internet" when there is temporary reception loss

3.2.5.5 Session Establishment

This section defines the procedure to pair and establish a CarPlay session over wireless, see Figure 3-3 (page 52) for details.

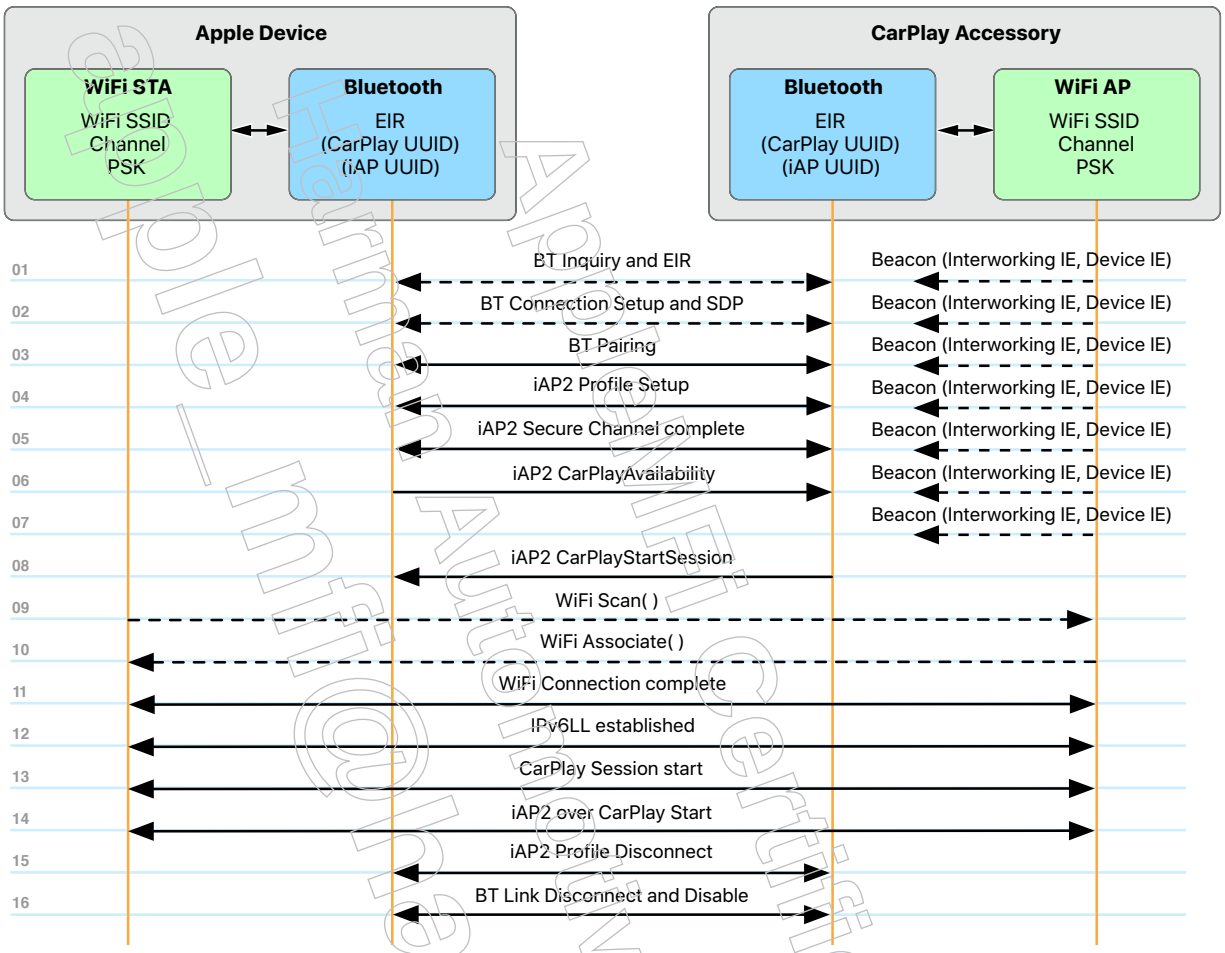

Figure 3-3: Initial Connection and Pairing using Bluetooth and iAP2

For first time connection establishment, the accessory must be discoverable over Bluetooth and allow for new devices to pair. If the accessory supports active Bluetooth pairing, it must use the Device CarPlay Bluetooth EIR to check if the device supports CarPlay, and if supported, offer the user to pair for CarPlay.

Once the pairing over Bluetooth is completed, the accessory must initiate an iAP2 session and wait for a minimum of 60 secs to receive a "15.6.1 CarPlayAvailability" (page 216) message. Once the message arrives and wireless CarPlay is available, the accessory must offer the user to connect for CarPlay, independent of how Bluetooth pairing was initiated. If CarPlay is not enabled on the device or a "15.6.1 CarPlayAvailability" (page 216) message is not received, the accessory must not offer CarPlay to the user and must proceed with pairing for classic Bluetooth.

Over iAP2 over Bluetooth, the accessory requests a session initiation using the "15.6.2 CarPlayStartSession" (page 217) message and provides the Wi-Fi credentials and the link-local IPv6 address to be used for CarPlay.

The accessory may delay the session initiation until the user has accepted to use CarPlay on the accessory. If the user does not accept within 60 seconds, the accessory may reconnect the device over classic Bluetooth.

Once the device establishes the session it will issue a disableBluetooth command (see "3.3.8.3 disableBluetooth" (page 163)) to trigger the accessory to disconnect the Bluetooth link to the device. See "3.2.5.9 Wireless Coexistence" (page 56).

At any point during setup or later, the user may turn off CarPlay over wireless on the device. In such situations, the accessory must initiate an iAP2 session over Bluetooth to receive an update of the "15.6.1 CarPlayAvailability" (page 216) message. If the "15.6.1 CarPlayAvailability" (page 216) message indicates that CarPlay is not available, the accessory must re-establish the connection with the device over classic Bluetooth.

Once a device has been paired and connected as a CarPlay device, the accessory must save it and be able to reconnect to it on subsequent system start-ups. Details on the reconnect procedures are available in "3.2.5.6 Session Reconnection" (page 53).

Accessories that support iAP2 over USB or CarPlay over USB must implement Out-Of-Band Bluetooth Pairing (see *Accessory Interface Specification*) to optimize the initial pairing procedure. After Out-Of-Band Bluetooth Pairing has completed, the accessory must store the device as the last used device on the system so it can be reconnected automatically after an ignition cycle.

3.2.5.6 Session Reconnection

This section defines the message flow to reconnect to a CarPlay-enabled device over wireless, see Figure 3-4 (page 53) for details.

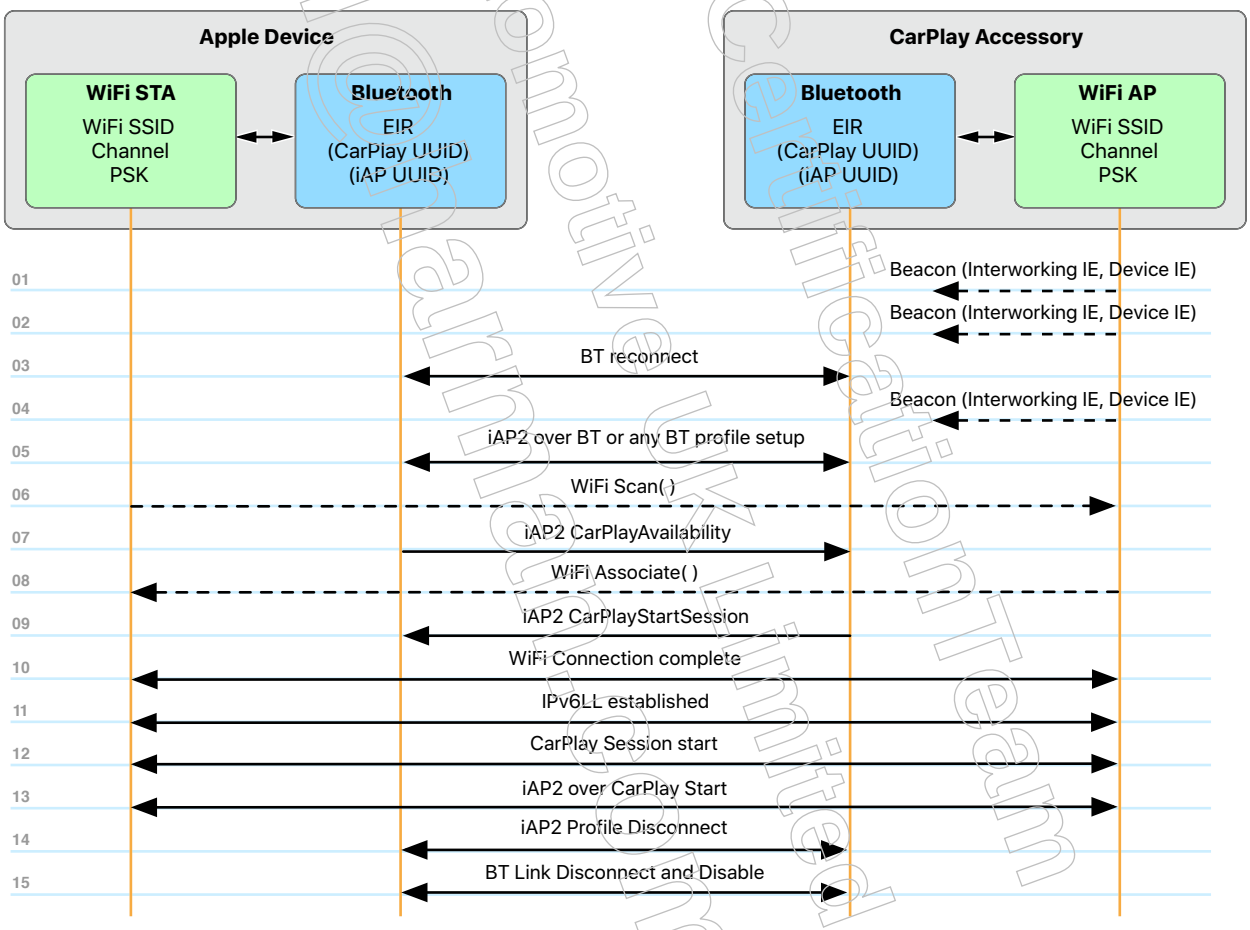

Figure 3-4: Reconnect using Bluetooth

The reconnection procedure is very similar to the first time connection, see "3.2.5.5 Session Establishment" (page 51), except that Bluetooth pairing already exists. The accessory must initiate reconnection to a previously paired device over Bluetooth.

Once the Bluetooth link is established, the device will scan and attempt to associate to the accessory's Wi-Fi access point using the previous Wi-Fi credentials. The accessory must connect iAP2 over Bluetooth to check that wireless CarPlay is enabled. If it is, the accessory must request a session initiation using the "15.6.2 CarPlayStartSession" (page 217) message. If CarPlay is not enabled, the accessory must reconnect the device over classic Bluetooth.

After the reconnection over Bluetooth, the accessory must follow the same procedure as during first time connection.

3.2.5.6.1 Reconnection Latencies

Users expect wireless CarPlay to be available as soon as the drive begins. In order to optimize the user experience, accessories must minimize startup latency for wireless CarPlay.

Wireless CarPlay startup latency can be measured in 2 phases: the startup connection request latency and the session start latency.

Startup connection request latency

The startup connection request latency for wireless CarPlay is defined as the period from explicit user action to put the car into a "ready to drive" state (e.g. pressing a "Start" button or turning the ignition on) to accessory transmission of the first Bluetooth connection packet after the Wi-Fi access point is ready to accept a connection. The startup connection request latency must not exceed 2 seconds.

In the following cases, the startup connection request latency may be as long as 7 seconds:

- If accessory UI and audio would remain active when the CarPlay session starts.
- If the accessory is powered on only after explicit user action (e.g. aftermarket or port install systems).

Session start latency

The session start latency for wireless CarPlay is defined as the period from accessory transmission of the first Bluetooth connection packet to the start of the CarPlay session. The session start latency must not exceed three seconds.

The overall time from system wake-up to start of the CarPlay session must not exceed 10 seconds. Accessories can speed up the reconnection process by connecting wireless CarPlay prior to the user action of setting the car in a "ready to drive" state (e.g. pressing a "Start" button or turning the ignition on) using cues such as the user approaching or unlocking the car, or opening a door.

For user initiated reconnections when the vehicle system has already booted up the reconnection latency is measured by the session start latency.

3.2.5.7 Session Termination

The accessory may disconnect an active CarPlay session in one of the following scenarios:

- "3.2.5.7.1 Leaving the Vehicle" (page 55)

3.2.5.7.1 Leaving the Vehicle

In this scenario, the accessory is being turned off as the user is leaving the vehicle.

In this case the accessory must disconnect the CarPlay session and then send a unicast Wi-Fi disassociation or de-authentication frame to all CarPlay clients.

The accessory's access point must not be turned off prior to the successful session termination.

The device will leave the Wi-Fi access point and subsequent reconnect will require a reconnect through Bluetooth.

3.2.5.7.2 Direct User Action

Disconnecting from the Accessory

In this scenario, the user is disconnecting an active CarPlay session or switching to another device from the accessory's UI.

In this case the accessory must disconnect the CarPlay session.

The device is expected to stay on the Wi-Fi access point and be ready for subsequent reconnects.

Unpairing or Deleting the Device from the Accessory

In this scenario, the user is unpairing or deleting a CarPlay device from the accessory's UI.

In this case the accessory must disconnect the session, delete any stored data about the device and may send a unicast Wi-Fi disassociation or de-authentication frame to disconnect the device from the Wi-Fi network.

Disabling Wi-Fi on the Accessory

In this scenario, the user disables the accessory's Wi-Fi access point. In this case the accessory must:

1. Prompt the user to notify them the CarPlay session will be disconnected if Wi-Fi is disabled
2. If the user accepts, disconnect the session
3. Send a unicast Wi-Fi disassociation or de-authentication frame to disconnect the device from the Wi-Fi network.

The accessory should only prompt the user to turn on Wi-Fi again when user tries to manually connect to CarPlay using the accessory's UI.

Disconnecting from the Device

In the case that iPhone leaves the accessory's access point because of user action on the device, iPhone will send a disassociation frame indicating that it is leaving the network. When the accessory receives such disassociation frame, it must terminate the CarPlay session immediately and avoid attempting to reconnect automatically. However, on the next ignition cycle, the accessory should attempt to reconnect CarPlay and not classic Bluetooth.

3.2.5.7.3 Out of Range

In this scenario, the user leaves the vehicle with an active CarPlay session.

The accessory must be able to detect that the device is out of range and then terminate the active CarPlay session.

If CarPlay is displayed on the screen or playing audio in the background, the accessory must terminate the session within 10 seconds. If CarPlay is idle and the user is actively using the native UI, the accessory must terminate the session within 30 seconds.

3.2.5.7.4 Unexpected Disconnect

In the event the device with the active CarPlay session is disconnected without a direct user action and is not detected as out-of-range, the accessory must enable the Bluetooth subsystem and attempt to reconnect to the device. This requirement is meant to re-establish the CarPlay session with a device when a sudden Wi-Fi disconnect occurs.

3.2.5.8 Supporting Multiple Devices

At any time, there is only one active CarPlay session. However, accessories may provide a user interface to select the active CarPlay device. See *CarPlay Design Guidelines* for details on managing multiple devices.

Upon ignition, the accessory must attempt to reconnect to either the last used device or to the preferred device that is chosen by the user. If neither is present in the vehicle, the accessory must attempt to reconnect to the previously paired devices until an available device is found.

3.2.5.9 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.

3.2.5.9.1 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 device has established a Wi-Fi connection and successfully initiated a CarPlay session, the Bluetooth

link must be terminated with the 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 device has established a Wi-Fi connection and successfully initiated a CarPlay session, the Bluetooth link must be terminated with the device. A Bluetooth connection and profiles may be established with other devices that are not engaged in a CarPlay session.

3.2.5.9.2 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 3-13 (page 57).

Table 3-13: 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 3-4 (page 35).

3.2.5.9.3 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.

3.2.5.9.4 Multiple Wi-Fi Access Points

If the accessory supports multiple Wi-Fi access points 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 security 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.

3.2.5.10 Supporting USB Data Ports

This section defines requirements for accessories supporting CarPlay over wireless that also have USB ports that support data transfer.

Accessories supporting CarPlay over wireless that also have USB ports that support data transfer must support Out-Of-Band Bluetooth Pairing as described in *Accessory Interface Specification*.

3.2.5.10.1 USB port supports CarPlay

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

Upon detection of a USB connection with a 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 the transport identifiers provided in the "15.6.1 CarPlayAvailability" (page 216) message.

Connecting to USB when no CarPlay session is active

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 "3.2.4.1 Role Switch" (page 28)).

The accessory must complete Out-Of-Band Bluetooth Pairing if requested by the device. After pairing completes, the accessory must keep any Bluetooth profile connections to the device disconnected and must not interrupt the active CarPlay session.

Connecting to USB during a CarPlay over wireless session

If there is an active CarPlay session over wireless the accessory can choose to either:

- Initiate USB role switch to start CarPlay, starting NCM and supporting the same iAP2 features it would normally support for a CarPlay over USB session (see "3.2.4.1 Role Switch" (page 28) and "3.2.4.5 Session Establishment" (page 31)).
- Authenticate and identify for iAP2 supporting PowerSourceUpdates, and Out-Of-Band Bluetooth Pairing. Using additional iAP2 features over USB is possible but not recommended.

The accessory must not set up Digital Audio over USB.

Disconnecting a CarPlay over USB session

For systems that support both CarPlay over USB and CarPlay over wireless, if the device is disconnected during a CarPlay over USB session:

- Do not initiate a CarPlay over wireless session for that device, or any other device already paired to the system.
- If that device is not paired for CarPlay over wireless the accessory may choose to connect Bluetooth legacy profiles.

3.2.5.10.2 USB port does not support CarPlay

Upon detection of a USB connection with a 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 the transport identifiers provided in the "15.6.1 CarPlayAvailability" (page 216) message during the previous session.

All CarPlay over wireless accessories that have USB ports capable of carrying data, but do not support CarPlay over USB, must support iAP2 with the following features:

- PowerSourceUpdates
- Out-Of-Band Bluetooth Pairing

Using additional iAP2 features over USB is possible but not recommended. The accessory must not set up Digital Audio over USB for a device connected to CarPlay over wireless.

If there is no active CarPlay session, the accessory must complete Out-Of-Band Bluetooth Pairing upon request by the device. After pairing completes, the accessory must initiate a CarPlay over wireless session and keep all Bluetooth profiles to this device disconnected during the session.

If there is an active CarPlay over wireless session with a device different than the device connected over USB, the accessory must not interrupt the active CarPlay session. See *CarPlay Design Guidelines* for details on managing multiple connected devices.

3.2.6 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 "3.3.9 CarPlay Communication Plug-in" (page 177)

3.2.6.1 iAP2 Client over USB

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

- *Accessory Authentication* as defined in the *Accessory Interface Specification*.
- "8 Addendum: Accessory Identification" (page 190) with `VehicleInformationComponent` and:
  - USBHostTransportComponent for CarPlay over USB.
  - BluetoothTransportComponent and WirelessCarPlayTransportComponent if the accessory also supports CarPlay over wireless.
- *Device Power (Lightning)* as defined in the *Accessory Interface Specification*.
- "4 CarPlay Connection" (page 180)
- "12 Addendum: Location Information" (page 196) and provide location information from GNSS and other sensor data.
- *Out-of-Band Bluetooth Pairing* as defined in the *Accessory Interface Specification* if the accessory also supports CarPlay over wireless.

The location information will be used to augment the 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; for example, to display call information on a vehicle instrument cluster display.

Accessories may also provide sensor data and other vehicle status to the device, see "7 Vehicle Status" (page 189). If the accessory supports any `RangeWarning` parameters, the accessory must not present a prompt to the user to find fuel or charging stations using the native navigation system while guidance is being provided by CarPlay.

3.2.6.2 iAP2 Client over Bluetooth

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

- *Accessory Authentication* as defined in the *Accessory Interface Specification*.
- "8 Addendum: Accessory Identification" (page 190) with `VehicleInformationComponent` and:
  - USBHostTransportComponent if the accessory also supports CarPlay over USB.
  - BluetoothTransportComponent and WirelessCarPlayTransportComponent for CarPlay over wireless.
- "4 CarPlay Connection" (page 180)

The iAP2 client over Bluetooth is used to check the availability of CarPlay and to request establishing a CarPlay session.

The iAP2 client over Bluetooth must be disconnected upon receiving the "3.3.8.3 disableBluetooth" (page 163) command after the CarPlay session is established.

The accessory must establish iAP2 over Bluetooth connection every time the device is reconnected to the accessory (for both CarPlay mode and legacy Bluetooth mode). This connection must be used to check if the device has wireless CarPlay enabled or disabled and update the availability within the accessory's user interface.

3.2.6.3 CarPlay Client

The CarPlay client must implement the logic described in "3.3.3 Resource Management" (page 128) for resource sharing between the accessory and device. It must also manage user events and the transfer of digital UI and audio between the accessory and the device.

Payload data for the CarPlay client may be encoded according to a variety of standards dependent on the content type, as described in "3.2.7 Media Types and Formats" (page 62), 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

3.2.6.4 iAP2 Client over CarPlay Client

For accessories that support CarPlay over wireless, the CarPlay Communication Plug-in enables iAP2 over CarPlay, see "3.3.8.14 iAPSendMessage" (page 171).

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

- *Accessory Authentication* in the *Accessory Interface Specification*.
- "8 Addendum: Accessory Identification" (page 190) with `VehicleInformationComponent`, `LocationInformationComponent`, and:
  - USBHostTransportComponent for CarPlay over USB (if CarPlay over USB is supported).
  - BluetoothTransportComponent and WirelessCarPlayTransportComponent for CarPlay over wireless.
- "12 Addendum: Location Information" (page 196) and provide location information from GNSS and other sensor data.

Accessories supporting CarPlay over wireless must provide location information using GNSS, see *Location: GNSS* as defined in the *Accessory Interface Specification*, as the 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 device, see "7 Vehicle Status" (page 189). If the accessory supports any RangeWarning parameters, the accessory must not present a prompt to the user to find fuel or charging stations using the native navigation system while guidance is being provided by CarPlay.

3.2.6.5 iOS Applications for CarPlay

If the accessory supports iOS applications for CarPlay, it must setup at least one, but preferably many, `SupportedExternalAccessoryProtocol` parameters in their "15.7.1 IdentificationInformation" (page 219) message, see "8 Addendum: Accessory Identification" (page 190).

The `SupportedExternalAccessoryProtocol` parameters for each EA protocol must be set as follows:

- `ExternalAccessoryProtocolName` must be a reverse-DNS style name that is sensible for that manufacturer (e.g. "com.mycompany.<protocol_name>"). It is recommended that protocol names are declared such that future iOS applications for CarPlay can target specific brands, vehicle models, vehicle types, etc.
- `ExternalAccessoryProtocolMatchAction` must be:
  - 0 (no action) if the application requires a communication protocol with the vehicle
  - 4 (no action and no communication protocol) if the application does not require a communication protocol with the vehicle
- `ExternalAccessoryProtocolCarPlay` must be included in `ExternalAccessoryProtocol` parameter group.

All transports involved in CarPlay must declare the same set of EA protocols.

3.2.7 Media Types and Formats

Video content (User Interface streams) will stream from the device on dedicated channels. One stream is used for the main display, while up to two further streams can be used for instrument cluster content. Two additional dedicated channels are required for audio: main audio (input and output) and alternate audio.

All control and video data packets are encrypted (ChaCha20). The task of encryption/decryption is abstracted for the accessory maker within the Apple-provided Communication Plug-in code.

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

3.2.7.1 User Interface Streams

A CarPlay user interface (UI) is streamed from the 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 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 for each display. See Table 3-14 (page 63). If the accessory supports both a main display and an instrument cluster display it must implement a decoder capable of decoding multiple H.264 streams in parallel.

The CarPlay UI content must always be rendered pixel for pixel without resorting to scaling or stretching. See "3.2.2.1 High Resolution Displays" (page 20) for display requirements.

The display resolution, max frame rate, physical dimensions and UI size(s) for CarPlay content for each display must be reported on connection via the "3.3.2.4 Info Message" (page 99).

Table 3-14: 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

Accessory makers wishing to offer resolution(s) requiring a H.264 profile greater than 4.2, or multiple displays with resolutions greater than 1920x720 must first consult with Apple for compatibility.

3.2.7.1.1 Main Display

It is strongly recommended to display the CarPlay content in a full screen configuration on the main display. The accessory may use a display with a resolution larger than 800 x 480 to present CarPlay content in a windowed configuration. In this case the accessory should transition between a windowed and full screen configuration, and do so only based on user interaction. Windowed configurations may have additional design requirements and must be reviewed by Apple.

The CarPlay UI must appear on the main 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.

3.2.7.1.2 Instrument Cluster Display

DEVELOPER PREVIEW

If an accessory has a high resolution instrument cluster display that presents navigation video content using its built-in navigation system, it must also display navigation video streams provided by CarPlay if available.

The accessory must negotiate support for instrument cluster UI content using the initial setup message. The device declares support by including an 'altScreen' string in the features array within the initial setup message request (see Table 3-64 (page 118)). If the device supports instrument cluster UI content, the accessory must include an 'altScreen' string in the enabledFeatures array within the initial setup message response (see Table 3-66 (page 119)). If the device does not support instrument cluster UI content, the accessory must not include an 'altScreen' string in the enabled-Features array within the initial setup message response.

The accessory is able to request up to two H.264 streams at one time (which are independent of the main display stream) to render different types of navigation UI content on the instrument cluster. The types of navigation UI content available to display are:

- Instruction card
- Map view
- Content determined by an iOS navigation app

Providing multiple H.264 streams for the instrument cluster display allows the accessory to render navigation UI content as different layers. For example, a map view could be used as a background with an instruction card rendered within a virtual tachometer on top of that background content.

The accessory must use the "3.3.8.23 showUI" (page 174) command to specify which type of navigation UI content will be rendered in a given instrument cluster UI stream. The showUI command is used to first generate CarPlay UI H.264 content in an instrument cluster display stream when it is not active, and to also change the content type when it is active. The accessory can optionally specify the visiblity of some map UI elements using the URL provided in the "3.3.8.23 showUI" (page 174) command.

The accessory must not send a showUI command upon connection. If instrument cluster content is required on connection the accessory must use the initialURL key in the display dictionary provided for that instrument cluster display surface in the info message response (see Table 3-36 (page 107)).

CarPlay UI must appear on an instrument cluster display within 500 ms of either:

- The accessory sending a "3.3.8.23 showUI" (page 174) command.
- The accessory receiving a session configuration that includes a screen stream of type 'Alt Screen', if an initialURL key is included in the display dictionary provided for that instrument cluster display surface in the info message response (see Table 3-36 (page 107)).

The accessory must not draw any navigation metadata, provided by CarPlay or the native navigation system, on top of navigation UI content provided by CarPlay. The accessory must not display any navigation metadata on an instrument cluster display that duplicates information already being displayed in CarPlay navigation UI content.

When CarPlay UI content being shown on the instrument cluster is removed from view, the accessory must send a "3.3.8.24 stopUI" (page 175) command.

The device can suggest to the accessory when relevant UI content is available to show on an instrument cluster display using the "3.3.8.25 suggestUI" (page 176) command. The content suggestions are provided in the form of an array of

the URLs described in "3.3.8.23 showUI" (page 174). The URL string provided in element 0 of the array is the primary suggestion, element 1 is the secondary suggestion and so on. If the device no longer suggests that certain content is relevant it will send another "3.3.8.25 suggestUI" (page 176) command with that URL removed. It is up the accessory to determine whether or not to use the "3.3.8.23 showUI" (page 174)) command to display any of the suggested content to the user.

In order to receive suggestions from the device the accessory must include the URLs for content types it supports in the altScreenSuggestUIURLs parameter in the info message response (see Table 3-32 (page 100)). The accessory must not register to receive suggestions for types of content it does not show to the user.

The device may support all, a subset of, or none of the navigation UI content types listed. The accessory must only use the URLs provided in altScreenURLs array in the info message request command (see Table 3-30 (page 100)) when populating the altScreenSuggestUIURLs parameter in the info message response and when using the "3.3.8.23 showUI" (page 174) command.

Note: Any application providing turn-by-turn guidance in CarPlay will set the Turn-by-turn Navigation app state (see "3.3.3.2 App States" (page 129)). Navigation applications available in CarPlay on the main display may not support instrument cluster content, so accessories are recommended to use the "3.3.8.25 suggestUI" (page 176) command instead of the Turn-by-turn Navigation app state (see "3.3.3.2 App States" (page 129)) in any logic used to determine when CarPlay UI content is shown on the instrument cluster.

If the accessory provides an interface for the user to increase or decrease the zoom level of a built-in navigation map shown on an instrument cluster display, it must do the same when displaying a navigation stream provided by CarPlay. The accessory must use the "3.3.8.28 changeMapZoomLevel" (page 177) command to notify the device when the user requests to zoom in or zoom out on the map. "3.3.8.28 changeMapZoomLevel" (page 177) must not be sent to the device when navigation video streams provided by CarPlay are not being displayed to the user.

3.2.7.1.3 Display Information

The accessory must describe each display surface that CarPlay UI will be shown to allow the device to configure the UI appropriately.

The accessory must report the horizontal and vertical dimensions of each display's glass panel in a `display` dictionary within the `displays` parameter of the info message response (see Table 3-36 (page 107) and Table 3-32 (page 100)).

For non-rectangluar displays the smallest rectangle that would fully enclose the display's glass panel must be declared. For example, see Figure 3-5 (page 66).

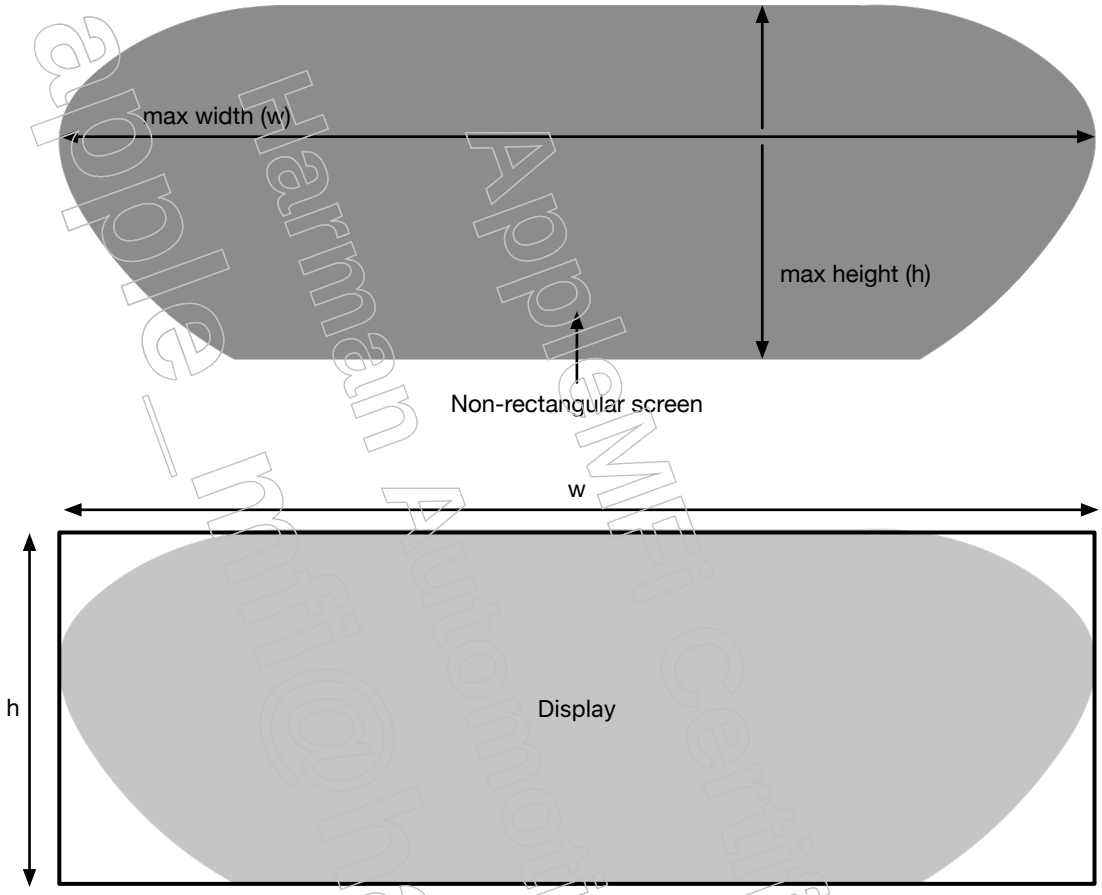

Figure 3-5: Non-rectangular display example

CarPlay UI content will be provided using an individual H.264 stream for each display surface. If two different types of navigation content are shown on an instrument cluster display at once they are considered as different display surfaces, so two alternate `display` dictionaries would be included in `displays` in the info message response.

The dimensions of a H.264 video stream setup by the device will match the resolution of the display it will be shown on.

Note: If the size of an instrument cluster display is significantly larger than the largest reported "3.2.7.1.4 View Area" (page 66) then the display dimensions reported to the device may be reduced to reduce the dimensions of the H.264 stream, if this significantly benefits accessory performance. Consult with Apple for guidelines on when a reduced display size can be reported.

3.2.7.1.4 View Area

A view area describes a rectangular area within a display that shows the CarPlay user interface (see Table 3-37 (page 109)). The CarPlay UI image will be drawn into a subsection of a H.264 frame determined by the size and position of the view area. The area outside of a view area will either be left unencoded or filled with black pixels. If no view area is contained within a view area, the device assumes the user can see and interact with every pixel of the display.

The position and size of the CarPlay image within a H.264 frame are identified in the VideoConfig packet sent in the video stream. The accessory must not display any H.264 video data outside of the area identified in the VideoConfig packet.

Any main display view area must be at least 800 x 480 pixels. Any instrument cluster view area must be:

- At least 200 x 100 pixels to show navigation instruction card content
- At least 300 x 200 pixels to show navigation map view content
- At least 300 x 200 pixels to show navigation content determined by an iOS navigation app

View Area Examples

Full Screen

The CarPlay UI is shown full screen on a 10 inch 1920 x 720 display. The accessory specifies identical `widthPixels` and `heightPixels` values in both the display (in black) and view area (in blue) for the full screen configuration.

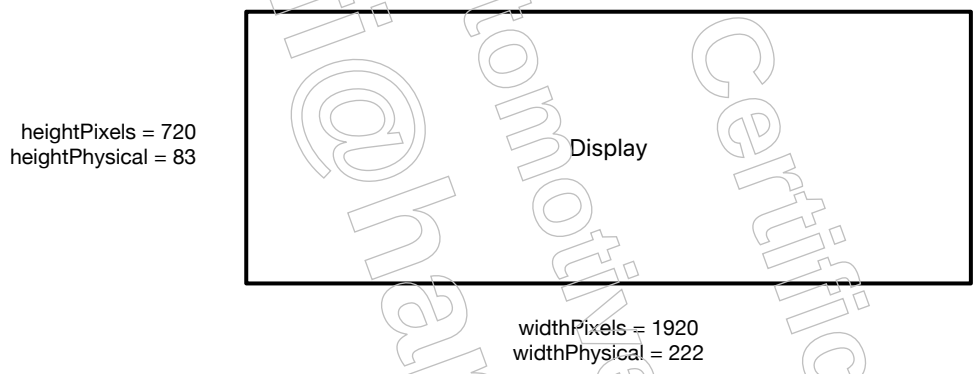

Figure 3-6: Display dimensions

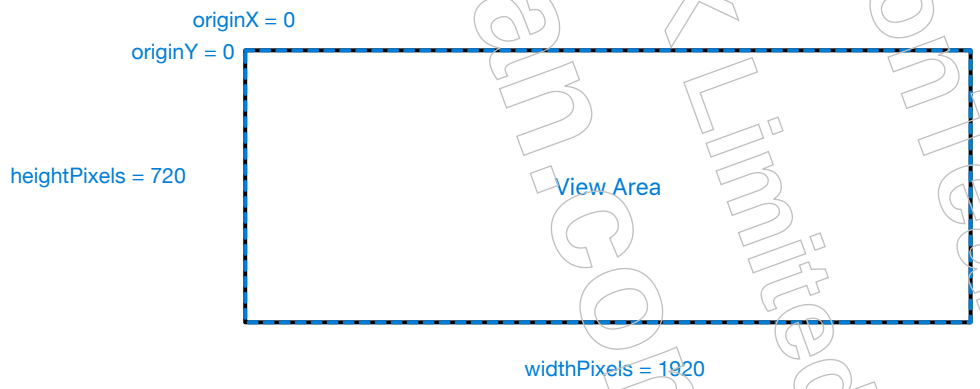

Figure 3-7: Full screen view area

Split Screen

The CarPlay UI is shown on the main display in a split-screen windowed configuration, alongside a native menu on the right hand side of the same 10 inch 1920 x 720 display. The accessory specifies a view area, with a smaller `widthPixels` value but the same `heightPixels`, along with the position of where that image should be shown relative to the top left corner of the display. As the native UI is shown on the right side of the display, the accessory would specify a view area with `originX` value of 0 so the CarPlay UI appears on the left.

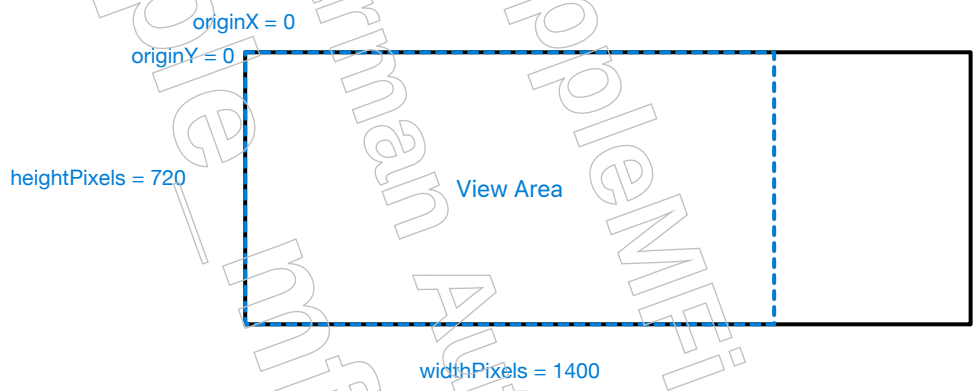

Figure 3-8: Split screen view area (left)

Instead, if the native UI is shown on the left side of the display, the accessory would specify a view area with `originX` value of 520 so the CarPlay UI appears on the right.

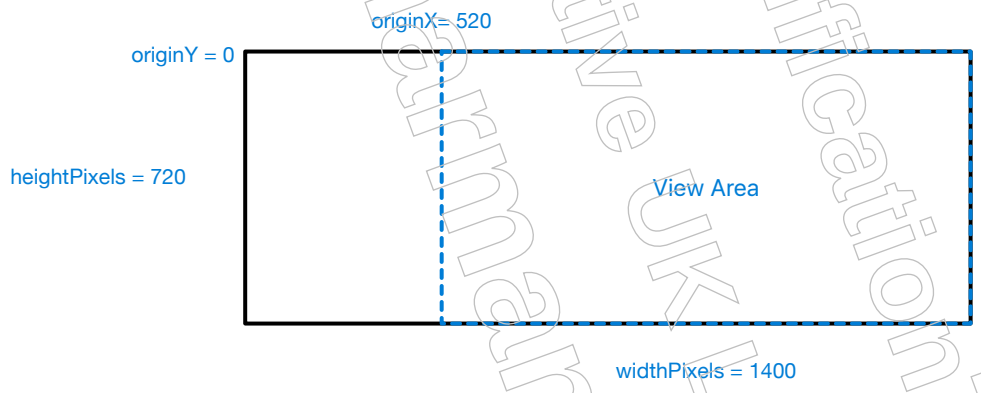

Figure 3-9: Split screen view area (right)

3.2.7.1.5 Safe Area

DEVELOPER PREVIEW

The term *safe area* refers to a subset of a view area in which every pixel is visible and, for touchscreen displays, is accessible to the user. One safe area can be declared per view area.

If no safe area is contained within a view area, the device assumes the user can see and interact with every pixel of that view area.

A safe area enables the accessory to inform the device if the CarPlay image is covered or obscured so that CarPlay may position critical information and user controls in areas that are visible to the user.

Safe areas can be used on both main and instrument cluster displays. When a safe area is used on a main display it may only be for the following reasons:

- Bezels placed on or adjacent to the display, blocking or obscuring the driver's view of the UI
- Non-rectangular displays

CarPlay UI will not be drawn outside of the safe area boundary on the main display by default. Drawing non-critical CarPlay UI outside of the safe area on a non-rectangular main display may be enabled using the drawUIOutsideSafeArea key (see "3-38 safeArea keys" (page 110)). Consult with Apple for guidelines on enabling drawing CarPlay UI outside of a safe area on a main display.

Any main display safe area must be at least 800 x 480 pixels. Any instrument cluster safe area must be:

- At least 100 x 50 pixels to show navigation instruction card content
- At least 200 x 100 pixels to show navigation map view content
- At least 200 x 100 pixels to show navigation content determined by an iOS navigation app

Non-critical CarPlay UI will be drawn outside of a safe area by default on an instrument cluster display. Using the drawUIOutsideSafeArea key (see "3-38 safeArea keys" (page 110)) is not supported for instrument cluster displays.

Safe Area Examples

Bezels On The Main Display

CarPlay is shown on a 8.8 inch 900 x 520 display that has a bezel fitted on top of it.

Figure 3-10: Display with bezel fitted

Information about the display (in black) would be specified in `widthPixels`, `heightPixels`, `widthPhysical` and `heightPhysical` in the `display` dictionary.

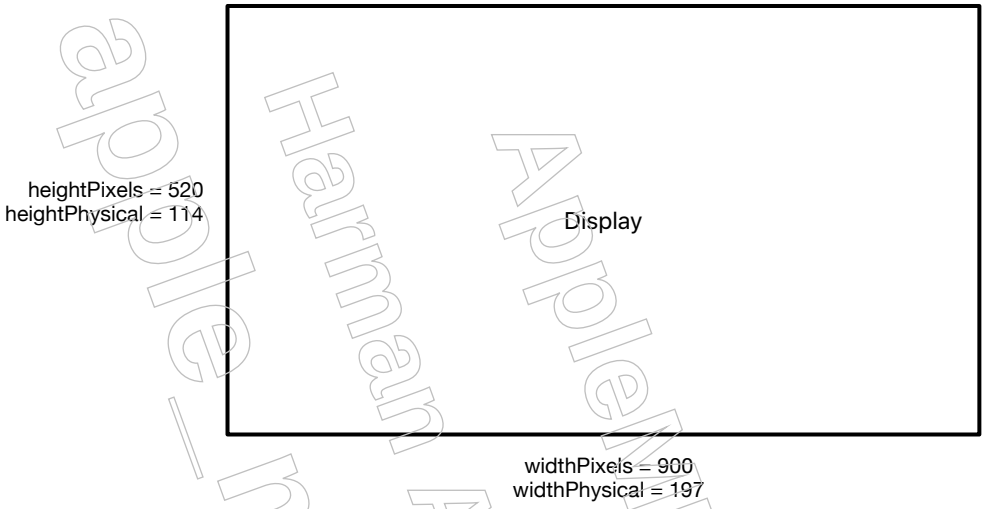

Figure 3-11: Display information

As the display is partly obscured by a bezel, the accessory defines a safe area (in green) within a view area (in blue) dictionary inside the `display` dictionary.

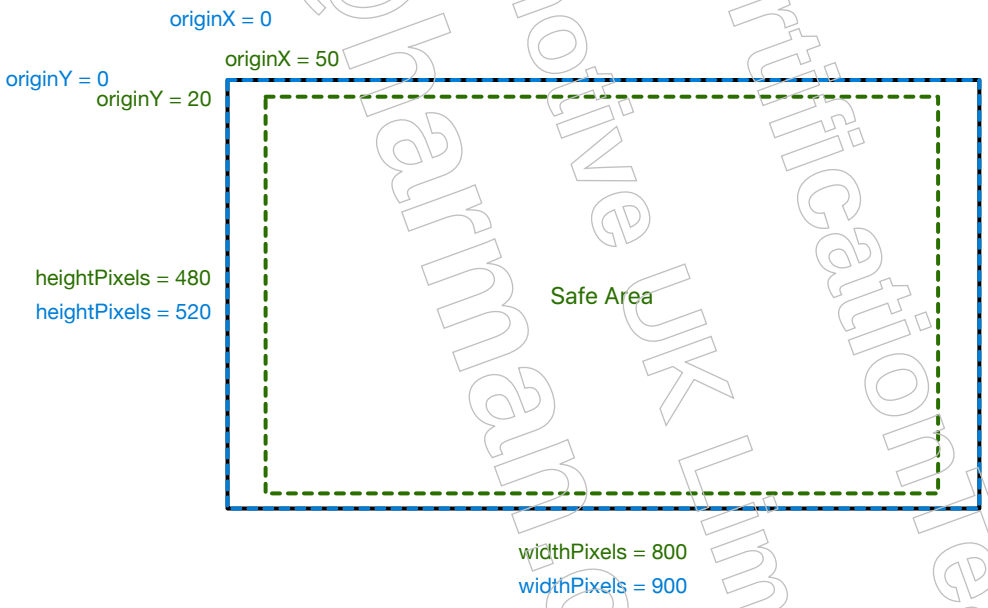

Figure 3-12: View area and safe area information

CarPlay will render all of the user critical elements of the UI within the safe area, so the user's view of those controls are not blocked by the bezel.

Figure 3-13: Unobscured safe area

Instrument Cluster

For an instrument cluster display the accessory may use a safe area to describe a visible area within a view area that is partially obscured by native UI elements.

For example, navigation content is shown between two circular virtual tachometers. The accessory declares a view area (in blue) for the navigation UI to be drawn into. The section of the UI that is fully visible to the user is identified in the safe area (in green), and CarPlay ensures the critical user information is contained entirely within that area.

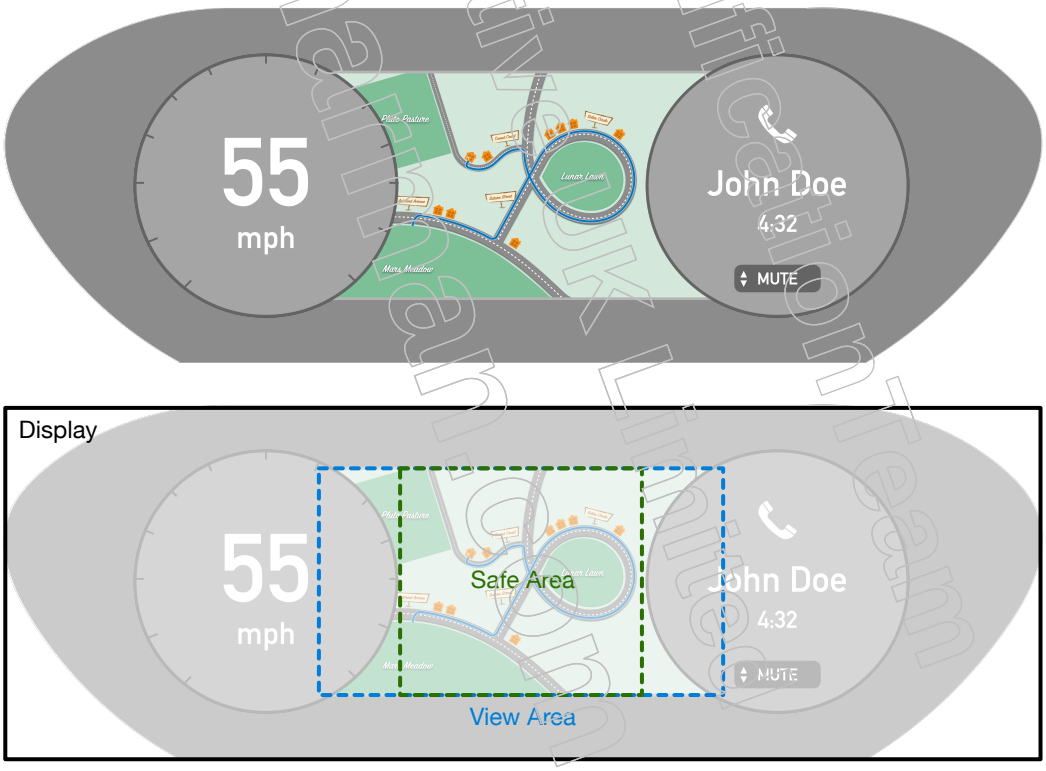

Figure 3-14: Safe area used within an instrument cluster display

3.2.7.1.6 UI Layout Configurations

CarPlay must participate in the UI configuration concepts available in the native accessory user experience. If each feature UI renders at the same fixed size (single UI layout configuration) then CarPlay must be shown to the user at this size. If the size and/or position of a native feature UI can be adjusted by the user (multiple UI layout configurations) then the CarPlay UI must be adjustable in the same way. This applies to both the main display and an instrument cluster display.

Single UI Layout Configuration

A single UI layout configuration describes when the CarPlay UI is shown in only one size and position on a given display.

If the CarPlay UI is presented within a windowed configuration on that display a single view area must be declared within the display information in the info message response, see Table 3-36 (page 107).

If the CarPlay UI is presented in a full screen configuration on that display a view area does not need to be declared in the display information. If a view area is declared the pixel dimensions must match that of the display.

Multiple UI Layout Configurations

A multiple UI layout configuration describes when the CarPlay UI is shown in multiple sizes and/or positions on a given display.

On connection the accessory must specify multiple view areas that represent different UI configurations for that display, and which configuration the system is in at the time of connection.

The accessory can request to transition between view areas to change the position and dimensions of the CarPlay UI as a result of user action at any time (see "3.3.8.18 requestViewArea" (page 172)). This allows the CarPlay UI to resize to match the current UI configuration without the need to reestablish the CarPlay session. If CarPlay is not being displayed the accessory must still update the view area whenever the UI configuration changes, so if CarPlay takes ownership of the screen the correct UI size is generated when the H.264 video stream is set up.

The accessory specifies how long a transition between two view areas should take when CarPlay is displayed on the screen. This allows the device to animate the CarPlay UI changing from one size and/or position to another, in a timeframe that matches the accessories own UI transition periods. The device uses the VideoConfig packet to update the size and position of the CarPlay UI during the transition, so the accessory can start and synchronize the native system user interface elements movements with CarPlay.

To change a safe area while maintaining a constant CarPlay image size, the accessory needs to declare multiple view areas with identical size and positions, but contain different safe area information, and use the requestViewArea command to transition between those view areas.

A transition control can be presented within the CarPlay UI on the main display for the user to request changing the user interface configuration. The transition control will be presented when `viewAreaTransitionControl` is set to True for a given view area (see Table 3-37 (page 109)).

The accessory must set `viewAreaTransitionControl` to True in order to show a transition control on the main display if either:

- a view area has identical dimensions to the display.

- there is no control outside of the CarPlay user interface to trigger a view area change.

If the transition control is pressed by the user, the device will send a "3.3.8.18 requestViewArea" (page 172) command to the accessory, which must respond with a "3.3.8.17 updateViewArea" (page 171) command to transition to the appropriate view area.

Showing a transition control in the CarPlay UI is not supported for instrument cluster displays.

An updateViewArea command must include an adjacentViewAreas array with exactly one element if the view area identified by viewAreaIndex has viewAreaTransitionControl set to True.

The transition control can only be used by accessories that support multiple UI layout configurations. A transition control must not be displayed by accessories that support a single UI layout configuration.

Legacy Display Mode

The accessory must check if the device supports view areas using the 'viewAreas' string in the features array in the setup message request (see "3.3.2.5 Setup Message" (page 117)). If the device does not support view areas, the accessory must not declare any view areas in the main display dictionary (see Table 3-36 (page 107)).

For a single UI layout configuration the accessory must use only the display dictionary in the info message response to generate a CarPlay UI image (that will fill all of the H.264 frame) at the desired size.

For multiple UI layout configurations only the display dictionary in the info message response must be used to generate a CarPlay image at the user preferred size, which would be the equivalent to the user preferred view area for devices that support view areas.

3.2.7.1.7 Corner Clipping Masks

When CarPlay is drawn into a H.264 stream it is possible the enclosing shapes within the CarPlay UI design can leave black areas in the corners of the stream. If displayed in a windowed configuration the black corner areas could compromise the visual integration of CarPlay into the native UI, especially if the native UI surrounding CarPlay is a non-black color.

To improve the visual integration in this configuration CarPlay can provide information about the shape of the black corner areas, in the form of corner clipping masks, so the accessory can set transparency levels for pixels in the CarPlay image for those areas, allowing a native background UI to be seen in place of the black corner areas.

The accessory must negotiate support for corner clipping masks in the initial setup message. The device declares support by including a 'cornerMasks' string in the features array within the initial setup message request (see Table 3-64 (page 118)). If the device supports corner clipping masks, the accessory must include a 'cornerMasks' string in the enabledFeatures array within the initial setup message response (see Table 3-66 (page 119)). If the device does not support corner clipping masks, the accessory must not include a 'cornerMasks' string in the enabledFeatures array within the initial setup message response.

When the device sets up a user interface video stream it will include a grayscale PNG image that is used to convey the clipping mask shape for the top left corner of the CarPlay UI (see Table 3-70 (page 122)). Each pixel in the PNG image provides a transparency value for a corresponding pixel in the CarPlay UI image. A white pixel in the PNG image represents a 0% transparent (opaque) pixel in the CarPlay image, whereas a black pixel represents 100% transparency. Gray

pixel values represent a transparency level between 0-100% to apply. For example, a 20% gray pixel would represent 20% transparency and a 75% gray pixel would represent 75% transparency.

The accessory must use the value of each grayscale pixel in the PNG image to set the correct transparency level for the corresponding pixel in the CarPlay UI image. The accessory must transpose the PNG image for use with each corner of the CarPlay UI, flipping the PNG image data horizontally for the top right corner, vertically for the bottom left corner and both horizontally and vertically for the bottom right corner.

The position and size of the CarPlay image within a H.264 frame are identified in the VideoConfig packet sent in the video stream. The accessory must use this information to determine the positions of the corners of the CarPlay UI when applying corner clipping masks. The accessory must not declare a safe area within any view areas (see Table 3-37 (page 109)) if it supports corner clipping masks.

For example, the accessory UI configuration view area is 1000 x 480 positioned at the origin of the display, the native UI has a blue background, the CarPlay UI is a red color and the device provides a 5x5 pixel PNG image to describe the top left corner shape. The process to blend the top left area of the CarPlay UI image over the native UI background is shown in Figure 3-15 (page 75).

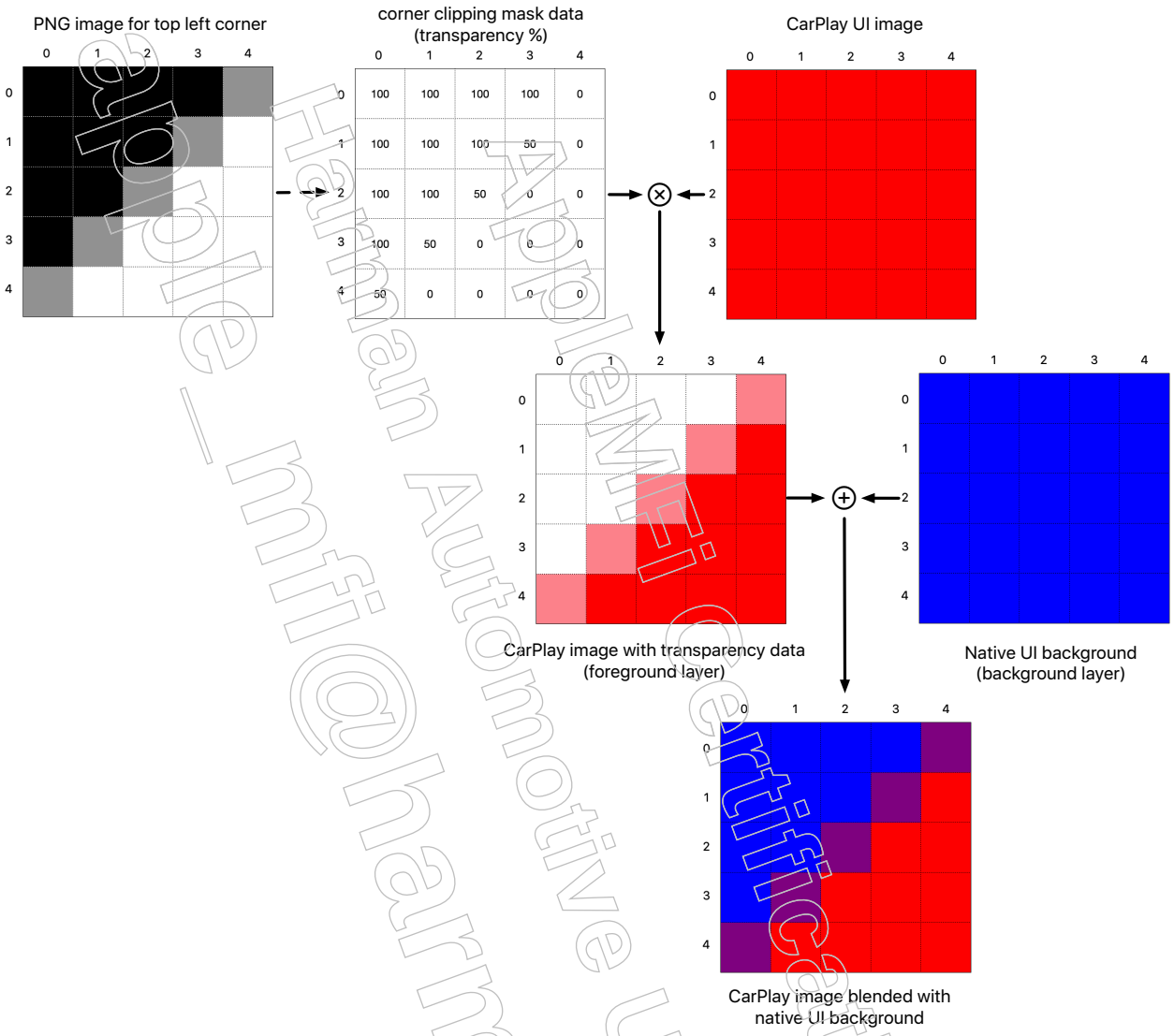

Figure 3-15: Top left corner blending using clipping mask

The process is similar for the bottom right area, as shown in Figure 3-16 (page 76).

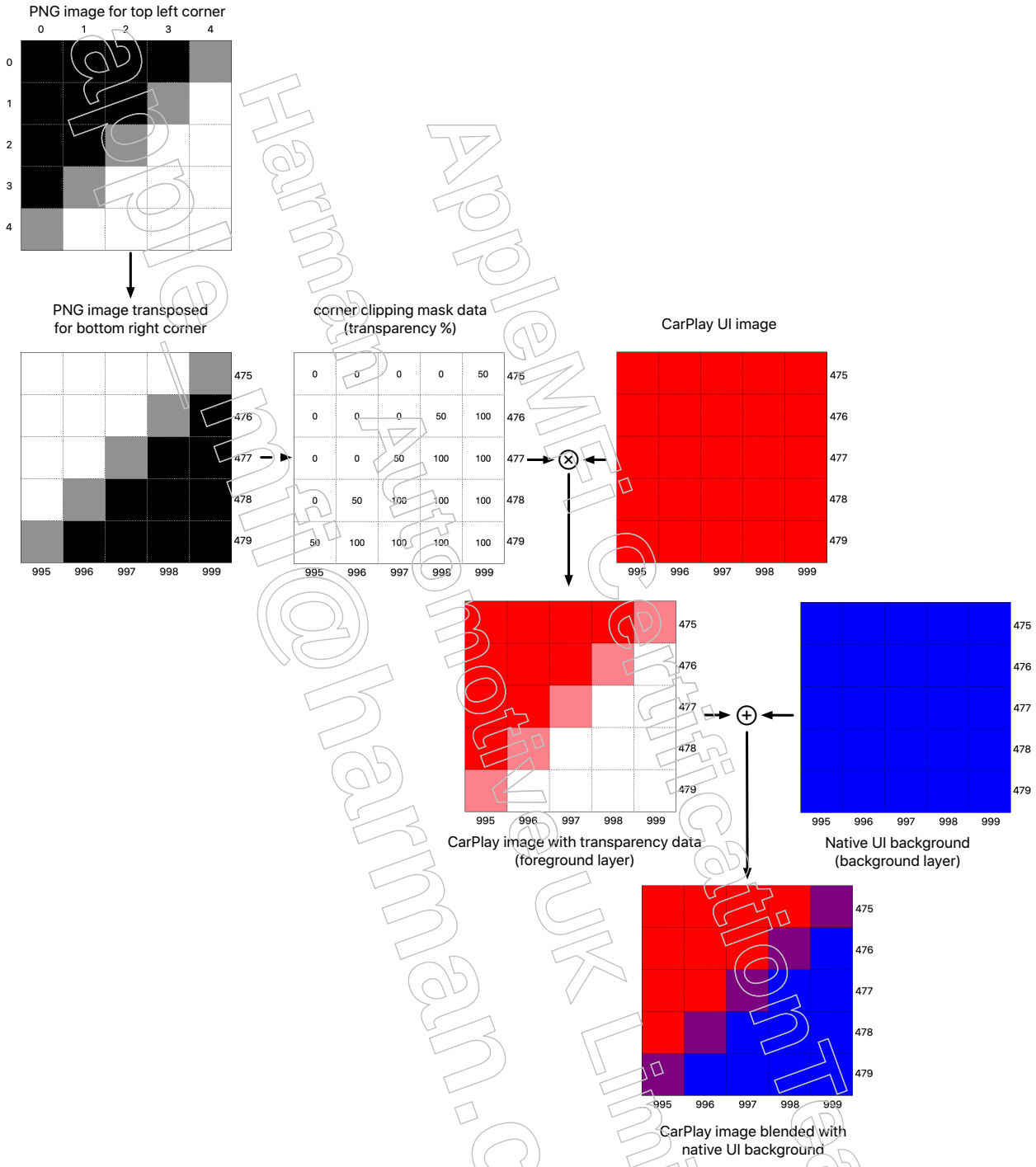

Figure 3-16: Bottom right corner blending using clipping mask

Outside of the corner areas the CarPlay UI pixels must remain entirely opaque (0% transparency). The accessory must use only the mask information provided in the PNG image included in the user interface stream setup message and not apply any other masking shapes to the CarPlay UI. Corner clipping masks must not be applied on instrument cluster displays.

3.2.7.1.8 Status Bar Position

The status bar in the CarPlay UI on the main display can be shown in a vertical configuration on the driver side (determined by the value of the rightHandDrive parameter in the info message response, see Table 3-32 (page 100), or in a horizontal configuration on the bottom of the UI. The device will determine where the status bar will be positioned based on the size of the CarPlay UI image being displayed.

For certain UI configurations the vehicle may want to specify the position of the status bar to improve the visual integration of CarPlay with the native UI. In this case, consult with Apple for guidelines on using the viewAreaStatusBarEdge parameter in the view area dictionary (see Table 3-37 (page 109)) to set the position of the status bar on the main display.

3.2.7.1.9 Appearance Modes

CarPlay can be presented to the user in different appearance modes. UI elements and maps in CarPlay can be either dark or light themed depending on user settings in the CarPlay UI, and other signals provided by the vehicle and within the device.

To improve the visual integration of CarPlay with the native system, the accessory can inform the device about the appearance of the native UI. For each display the accessory can describe the appearance of its system-wide UI, and separately describe the appearance of its map (if supported). The accessory can also report how the appearance mode is determined for each display:

- The appearance is set permanently light or dark by the system and can not be controlled through a user setting.
- The appearance changes dynamically when transitioning between day and night.
- The user has opted to set the appearance to be either light or dark through a user setting.

The accessory must set the values of uiAppearanceMode and uiAppearanceSetting in each display (see Table 3-36 (page 107)) declared in the info message response.

If the accessory can show a map on a display it must set the value of mapAppearanceMode and mapAppearanceSetting for that display (see Table 3-36 (page 107)) when declared in the info message response. It must not include mapAppearanceMode and mapAppearanceSetting for any displays that can not show a map.

During a CarPlay session, for each display:

- If the appearance of the system-wide UI changes the accessory must send a "3.3.8.26 uiAppearanceUpdate" (page 176) command to update the current UI appearance mode.
- If the user changes a setting that controls the system-wide UI appearance the accessory must send a "3.3.8.26 uiAppearanceUpdate" (page 176) command to update the current UI appearance setting.
- If the appearance of the map changes the accessory must send a "3.3.8.27 mapAppearanceUpdate" (page 176) command to update the current map appearance mode.
- If the user changes a setting that controls the map appearance the accessory must send a "3.3.8.27 mapAppearanceUpdate" (page 176) command to update the current map appearance setting.

3.2.7.2 Audio

DEVELOPER PREVIEW

The accessory must support audio input and output streams encoded as follows:

- CarPlay over USB
  - LPCM.
- CarPlay over wireless
  - Raw AAC-LC for high latency audio (main high audio).
  - OPUS for low latency audio (alternate audio, auxiliary audio and main audio except "media").

Accessories supporting CarPlay over wireless must support multiple decode instances and concurrent decode/encode instances.

Volume control and display of volume control UI are the responsibility of the accessory because the device provides only unattenuated line-level audio output. See "3.2.7.2.12 Volume Management" (page 90) for specific requirements and recommendations.

The accessory must support four independent audio streams - main (input and output), alternate, auxiliary input and output. The main audio stream from the device is using a shared resource with the accessory as described in "3.3.3.1 Resources" (page 128). Alternate and auxiliary output audio must be mixed with main audio regardless of whether the source of main audio is the device (for example, the iOS Music app) or the accessory (for example, an FM tuner on the vehicle head unit), see "3.2.7.2.11 Mixing" (page 90). In some use cases output on alternate and/or auxiliary output might be accompanied by a request to duck the main audio channel ("3.2.7.2.13 Ducking" (page 91)).

The auxiliary input and auxiliary output audio streams are set up by the device as needed, when Siri is listening or speaking back to the user. The accessory must be capable of setting up the auxiliary audio streams at any time, upon request from the device. The device may request to set up one or both streams.

The audio sample rates supported by the accessory must be reported using the Info message, see "3.3.2.4 Info Message" (page 99). The accessory must only support sample rates that are required or recommended for that audio format, and must only report sample rates that are natively supported in hardware.

Accessories that implement the CarPlay feature must respect the requirements from *Digital Audio: Multiple Audio Connections* as defined in the *Accessory Interface Specification*. In addition, they must use the CarPlay audio stream when using the feature and must not use another transport such as USB Host Mode Audio or Bluetooth in parallel.

The device will at times simultaneously employ different sample rates for the main (input and output), auxiliary and alternate audio streams, or may require sample rate transitions, for instance when transitioning main audio from music to a phone call. Main, auxiliary 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 device using the "3.3.2.5 Setup Message" (page 117).

Also, the 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 device sends a "3.3.2.5 Setup Message" (page 117) 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 acknowledgment of the setup message. For input the accessory must be ready to send audio samples to the device after the acknowledgment of the setup message.
- It is ready to begin to continuously receive audio from the device (if main or alternate audio output was requested). For output, the accessory must output all samples given to it after an acknowledgment of the setup message.

The same requirements also apply to auxiliary audio streams for accessories that support them, however the accessory must reply to the setup message within 50 ms.

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 16 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 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 device should be bit-identical to that of an audio signal sent from other sources, assuming those sources allow direct digital transfer.

3.2.7.2.1 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 3-15 (page 79) 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 3-34 (page 105) for a list of audio configurations.

Table 3-15: Entertainment Stream Requirements

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

3.2.7.2.2 Main Audio - Telephony

DEVELOPER PREVIEW

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*, *ITU-T P.1110: Wideband hands-free communication in motor vehicles* and *ITU-T P.1120: Super-wideband and fullband hands-free communication in motor vehicles*, whichever applies to the current use case.

Proof of compliance with the *ITU-T P.1100*, *ITU-T P.1110* and *ITU-T P.1120* specifications, where applicable, must be submitted to Apple as part of the self-certification audit process.

Sample Rate

Table 3-16 (page 80) details the audio stream requirements. The accessory must accept and provide only the required sample rates with equal or higher bit depth.

Table 3-16: Telephony Audio Stream Requirements

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

As an optimization, the device may prefer to encode the audio stream at a sample rate higher than the one used by the cellular network to encode a call. The accessory may use the vocoderInfo parameter in the setup message to optimize telephony audio playback for the native sample rate of the call (see Table 3-67 (page 120)). Using vocoderInfo is required when the accessory supports CarPlay over wireless as not all sample rates used by cellular networks are supported by the audio encoders used with CarPlay over wireless.

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. The echo cancellation must ensure DTMF tones played through the cabin speakers of the vehicle are not present in the uplink audio path.

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-T measurement procedure.

Speech Quality in Noise

The requirements for measuring speech quality in noise differ by vehicle type:

- Passenger vehicles may take background noise recordings from the vehicle directly or use standardized recordings from the ITU-T database. Refer to *ITU-T Telephony Audio Quality Tests* for more information.
- Alternative vehicle types (e.g., trucks, tractors, motorcycles) must use background noise recordings collected in the vehicle being certified. Motorcycles using headsets for communication must record vehicle noise at the headset microphone location.

Refer to Section 9.2.1 of *ITU-T P.1100*/*ITU-T P.1110* and Section 8.1.2 of *ITU-T P.1120* for recording procedure.

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 3-17 (page 81).

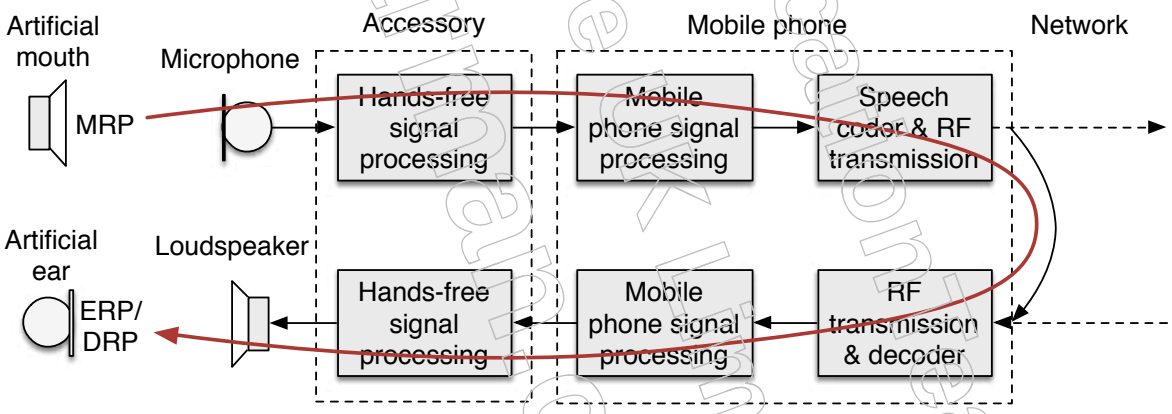

Figure 3-17: Telephony Audio Round-Trip Path

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, *ITU-T P.1110*, Clause 11.4.1 for wideband telephony and *ITU-T P.1120*, Clause 9.4.1 for super-wideband telephony.

3.2.7.2.3 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 3-17 (page 82) details the audio stream requirements. The accessory must accept and provide the following sample rate with equal or higher bit depth.

Table 3-17: 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-T measurement procedure.

Speech Quality in Noise

Refer to Section 9.2.1 of *ITU-T P.1100*/*ITU-T P.1110* and Section 8.1.2 of *ITU-T P.1120* for recording procedure (see "3.2.7.2.2 Main Audio - Telephony" (page 80)).

Round-Trip Delay

The round-trip delay must meet the same requirements as defined for Telephony (see "3.2.7.2.2 Main Audio - Telephony" (page 80)).

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 3-18 (page 83). The mask (see Figure 3-18 (page 84)) 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 3-18: 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 3-18 (page 84)

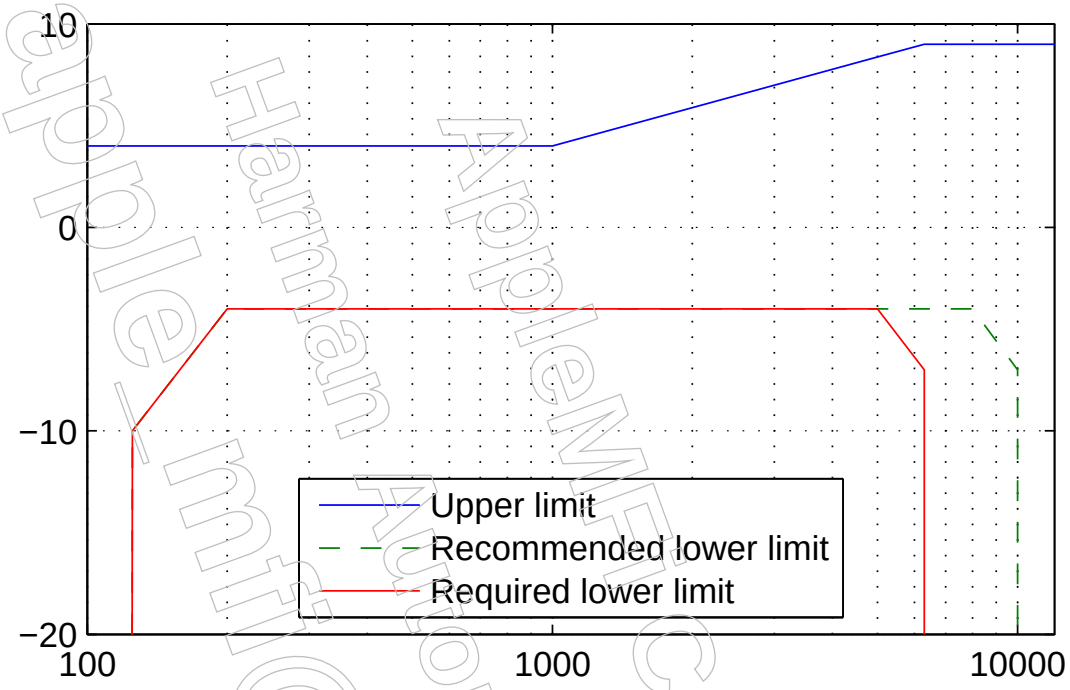

Figure 3-18: Audio Tolerance Mask For Send Frequency Response (Sensitivity (dB) vs. Frequency (Hz)

3.2.7.2.4 Main Audio - Speech Recognition

Note: Main Audio with audio type "speechRecognition" will not be used for Siri, but it will be used for other iOS applications that require a duplex audio stream.

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

Sample Rate

Table 3-19 (page 84) details the audio stream requirements. The accessory must accept and provide the following sample rate with equal or higher bit depth.

Table 3-19: Speech Recognition 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-T 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 speech recognition must meet the same requirements as defined for FaceTime. See "3.2.7.2.3 Main Audio - FaceTime Audio" (page 82)

3.2.7.2.5 Main Audio - Alert

The alert audio type is a low latency audio stream.

Sample Rate

Table 3-20 (page 86) 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 3-34 (page 105) for a list of audio configurations.

Table 3-20: 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)
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.

3.2.7.2.6 Main Audio - Default

The default audio type is a duplex audio stream.

Sample Rate

Table 3-21 (page 86) details the audio stream requirements. The accessory must accept and provide the required sample rates with equal or higher bit depth.

Table 3-21: 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 "3.2.7.2.2 Main Audio - Telephony" (page 80) and "3.2.7.2.3 Main Audio - FaceTime Audio" (page 82).

3.2.7.2.7 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 3-22 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 3-34 (page 105) for a list of audio configurations.

Table 3-22: Main High Audio Stream Requirements

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

3.2.7.2.8 Alternate Audio

Sample Rate

Table 3-23 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 3-34 (page 105) for a list of audio configurations.

Table 3-23: 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.

3.2.7.2.9 Auxiliary Input Audio - Speech Recognition

DEVELOPER PREVIEW

The auxiliary input audio stream is an input only stream used for Siri.

Sample Rate

Table 3-24 describes requirements for the audio input stream. The accessory must accept and provide the following sample rates with equal bit depth.

Table 3-24: Auxiliary Input Audio Stream Requirements

Sample Rate	Bit Depth	Channels	Duplexing	Requirement
16 kHz	16 bit	mono	n/a	Required

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, P.1100, or P1120. 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 receipt of sound at the accessory's microphone to the receipt of a buffer of LPCM audio by the CarPlay Communication Plug-in representing that sound must not exceed 60ms.

Microphones

Directional microphones and linear beam-forming microphone arrays are recommended for improved SNR.

Echo cancelation and noise reduction

Echo cancelation and noise reduction algorithms must be tuned to prevent distortions to the original uplink speech signal. Linear echo cancelation 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. These algorithms must not impact the trigger and word error rate accuracy.

Echo cancelation must be applied to the recorded microphone input to compensate for any audio played through the accessory speakers. This includes all CarPlay audio played using Main Audio or Alternate Audio, native media playback and notifications.

Buffering

"Microphone data is recorded, processed and stored continuously on the accessory in a circular buffer as outlined in "3.2.2.4 Speakers and Microphone" (page 21). The size of the circular buffer is configured by the device using the bufferSizeMs parameter in Table 3-79 enhancedSiriParameters keys (page 127) sent using SET_PARAMETER, see Table 3-78 SET_PARAMETER method keys (page 126).

Whenever an auxiliary input audio stream is setup by the device, a streamStartTimestamp parameter will be included (see Table 3-67 Audio stream descriptors request keys (page 120)). The accessory must send some of the stored circular buffer data starting at streamStartTimestamp, as well as continue streaming real time audio input as the driver is speaking. Audio data must be sent to the device at burst intervals which are preconfigured by the device using burstPeriodMs in the SET_PARAMETER method (see Table 3-78 SET_PARAMETER method keys (page 126))."

3.2.7.2.10 Auxiliary Output Audio - Speech Recognition

The auxiliary output audio stream is an output only stream used for Siri.

Sample Rate

Table 3-25 describes requirements for the audio output stream. The accessory must support only 48 kHz 16-bit audio natively, that is, a sample rate at which the accessory can perform its audio processing without sample rate conversion.

Table 3-25: Auxiliary Output Audio Stream Requirements

Sample Rate	Bit Depth	Channels	Duplexing	Requirement
48 kHz	16 bit	mono/stereo(*)	n/a	Required (*) stereo supported for CarPlay over USB only.

Latency

The elapsed time 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.

3.2.7.2.11 Mixing

DEVELOPER PREVIEW

Both accessory and device must distinguish between main, auxiliary and alternate audio, and route audio accordingly. Figure 3-19 (page 90) illustrates conceptually the various audio sources and shows how they must be mixed.

Accessories must be able to mix auxiliary output audio with all audio played through the accessory speakers, regardless of the source of audio. This includes any main audio type from the device, such as media, alert, phone call, speech recognition, or audio from the accessory, such as FM tuner, as well as alternate audio or other notifications from the native system.

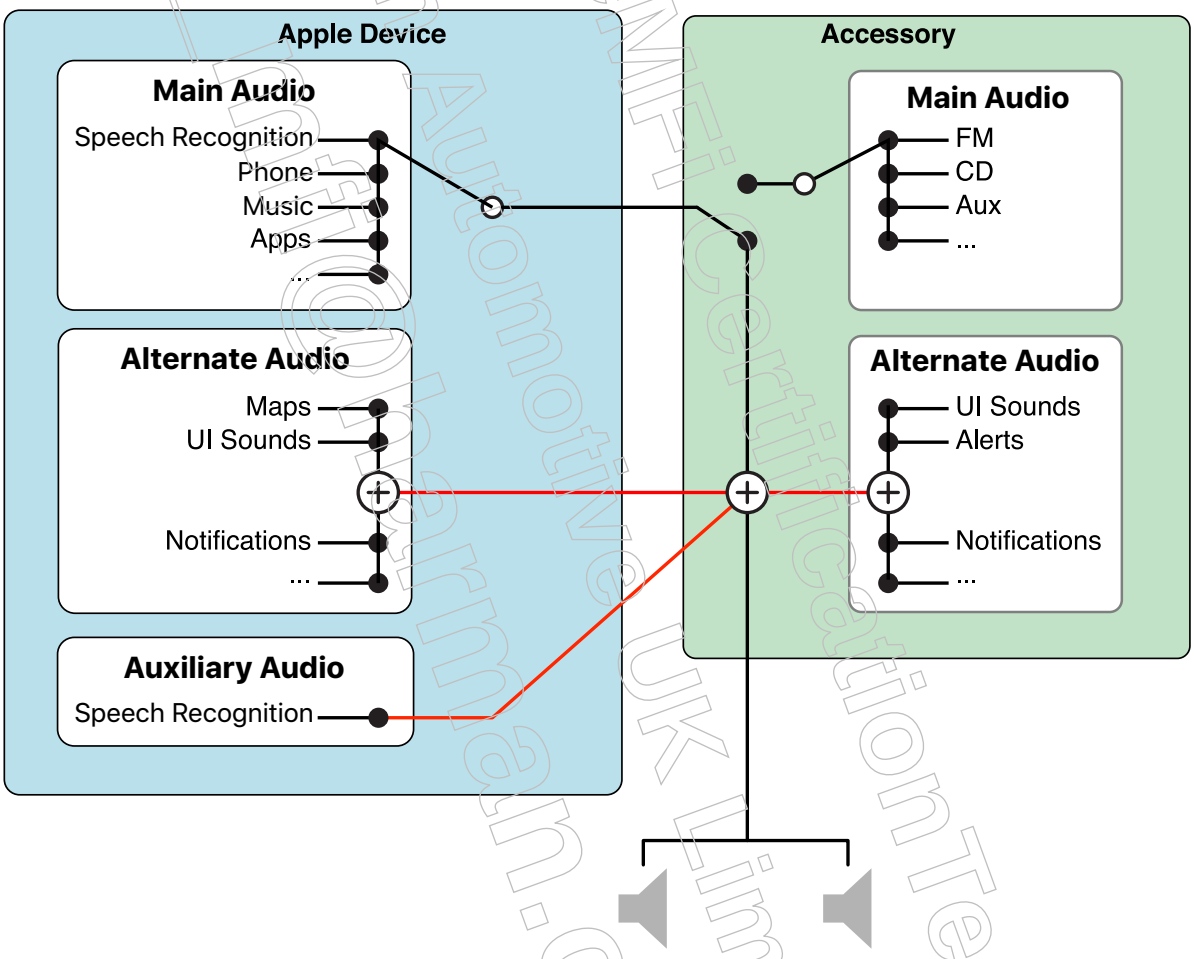

Figure 3-19: Mixing CarPlay audio

3.2.7.2.12 Volume Management

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

If the accessory supports maintaining separate volumes for different types of audio, it must do so using the following

volume categories, with corresponding mappings to main audio stream audioTypes (see Table 3-73 (page 123)), and alternate audio:

- Entertainment - main audio `media`
- Ringtone - main audio `alert`
- Phone Calls - main audio `telephony`
- Voice Prompts and Notifications
  - main audio `speechRecognition`
  - auxiliary output audio `speechRecognition`
  - 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".

If the user interacts with the accessory's primary volume controls (e.g. steering wheel buttons, volume knob) the accessory must adjust the volume of active CarPlay audio streams as follows:

- If there is an active CarPlay alternate audio stream the volume control must adjust only the level of this stream between duckAudio and unduckAudio commands (see "3.2.7.2.13 Ducking" (page 91)).
- If there is an active CarPlay auxiliary output audio stream the volume control must adjust only the level of this stream between duckAudio and unduckAudio commands (see "3.2.7.2.13 Ducking" (page 91)).
- If both an auxiliary output audio stream and alternate audio stream are active, the volume control must adjust only the level of the auxiliary out audio stream between duckAudio and unduckAudio commands (see "3.2.7.2.13 Ducking" (page 91)).
- If there is an active CarPlay main audio stream, and the volume control is not adjusting alternate or auxiliary out audio streams as described above, the volume control must adjust the level of the main audio stream.

3.2.7.2.13 Ducking

DEVELOPER PREVIEW

The accessory must support ducking, the process of ramping the main media audio volume down so the user can more clearly hear an announcement or alert from the alternate audio channel or Siri interactions from the auxiliary audio channels. 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) "Arriving at your destination" can be played over alternate audio. Another example is when the user starts interacting with Siri the main audio level may be reduced so the user can talk to Siri and hear feedback over the auxiliary audio channels. Ducking fades the main media 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.

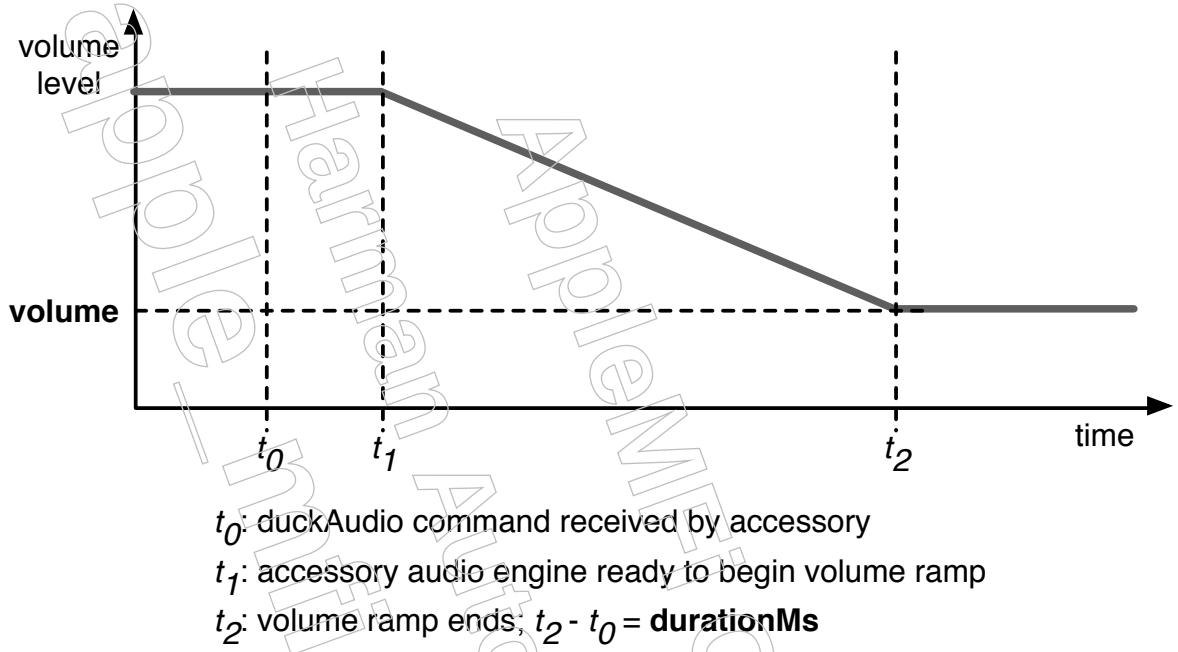

Figure 3-20: Event timeline for audio ducking/unducking

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.

The accessory must only duck the main audio volume if it receives an explicit `duckAudio` command from the device, and must only unduck the main audio volume if it receives an explicit `unduckAudio` command from the device. Other commands (e.g. audio setup or teardown commands) must not be used to initiate ducking or unducking of main audio volume.

3.3 CarPlay Communication Protocol

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

- Setup and control audio and video (UI) streaming
- Arbitrate ownership and use of the accessory's shared resources
- Communicate the 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.

3.3.1 Discovery

Deprecated: Discovery using Bonjour and CarPlay Control are deprecated and replaced by "4 CarPlay Connection" (page 180).

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

3.3.1.1 Device Discovery by Accessories

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 a 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: 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 "3.3.1.1.1 CarPlay Control" (page 93).

The name of the Bonjour service is the user-visible name of the 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 3-26 (page 93).

Table 3-26: CarPlay Control Bonjour Service Keys

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

3.3.1.1.1 CarPlay Control

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

- Look up the DNS name of the device with Bonjour by resolving the _carplay-ctrl._tcp service
- Resolve the DNS name to the IP addresses of the CarPlay Control device. If the accessory platform's `getaddrinfo` is Bonjour aware then it may be used. Otherwise, Bonjour's asynchronous DNSServiceGetAddrInfo API should be used.
- Connect to the CarPlay Control device's IP addresses. 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 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 a device that is in a sleep state.

- 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 3-27: CarPlay Control Commands

Command	Description
connect	Requests that the 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 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 device to initiate a CarPlay session to it then it would send the following HTTP request to the device:

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

3.3.1.2 Accessory Discovery

CarPlay accessories must advertise a Bonjour service to the device. The advertised service must be of a server type `_airplay._tcp`. 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 3-28 (page 95). The CarPlay Communication Plug-in will interface to the accessory's Bonjour implementation to configure and broadcast these values.

Table 3-28: 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 identify the acoustic environment of the product. The model name should include a unique identifier/code name of the vehicle brand, vehicle model, and a label for the unique acoustic tuning applied to the system.
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 3-47 (page 113).
pi	Y*	Pairing identity string. See "3.3.2.1 Pairing" (page 97). * Required when the accessory supports CarPlay over wireless.

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.

3.3.2 Setup and Control

When the accessory sends a "15.6.2 CarPlayStartSession" (page 217) message to a device, the device initiates the connection to the accessory using the provided parameters and starts a CarPlay session. HTTP/RTSP is used to set up, control, and monitor a CarPlay session.

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 "3.2.7 Media Types and Formats" (page 62).
- Scheduling the video signal to a synchronized clock between both accessory and 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 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.

The following summarizes a typical session lifetime:

1. The accessory sends a "15.6.2 CarPlayStartSession" (page 217) message to the device.
    - The device connects to the accessory using the provided port.
2. The device authenticates the accessory and sets up a secure channel.

- This process requires the accessory to perform the message exchange described in "3.3.2.1 Pairing" (page 97) and "3.3.2.2 Encryption" (page 97).
- After this step, all subsequent control requests and responses are encrypted.

3. The device sends an info request to provide its info and get the accessory's info.
   - The request message is a plist providing the device's model, version, etc.
   - The response message is a plist returning the accessory's audio formats, display info, model, current modes, etc.
   - See "3.3.2.4 Info Message" (page 99).
4. The 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 device will start listening for UDP packets.
   - For output UDP streams, the accessory will start listening for UDP packets.
   - See "3.3.2.5 Setup Message" (page 117).
5. The device makes a TCP connection to the accessory for each TCP-based stream.
   - For both input and output streams, the device initiates the connection.
   - For output streams, the device sends data to the accessory (e.g., music to play on the accessory).
   - For input streams, the accessory will send data to the device (e.g., HID reports). See Table 3-67 (page 120) and Table 3-70 (page 122)).
6. The device sends a record request to start a session on the accessory.
   - The accessory performs time sync negotiation with the device. See "3.3.2.9 Synchronization" (page 127).
   - The accessory starts corresponding streams, creating threads to process audio/video data, timers, etc.
7. The device sends setup requests to set up streams for audio and/or screen, and makes TCP connections as needed.
8. The device and accessory exchange data.
   - The device sends audio and video output data to the accessory for playback.
   - The accessory sends audio input data (if any) to the device for input.
9. The device sends a TEARDOWN request to end one or more streams on the accessory.
   - The accessory stops processing for specified streams and releases corresponding resources.
10. The 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 device closes all TCP connections to the accessory.
    - This resets the device back to the point before it started the session.

3.3.2.1 Pairing

Pairing establishes a device-to-accessory association, saves it to a persistent storage, and uses it on subsequent sessions to verify authenticity and create encrypted communication channels. If the device and the accessory are not paired, pair-setup must be performed. CarPlay currently uses rudimentary "unauthenticated" pair-setup with the fixed PIN provided by the device on behalf of the user.

Pairing requests are performed by sending a POST request to the accessory's HTTP server. The URL resource specifies the pairing operation to perform (`/pair-setup` or `/pair-verify`), see "3.3.2.8 URLs" (page 125). The body of the HTTP request contains the TLV data. Pairing responses are delivered as TLV data in the body of the HTTP response. The MIME type for requests and responses is "`application/pairing+tlv8`".

3.3.2.2 Encryption

The control and event channel and media payloads utilize differing encryption mechanisms.

3.3.2.2.1 Control and Event Channel Encryption

All control communication channels become fully encrypted once the pair-verify process completes successfully.

The following key derivation scheme is used for the control connection:

ControlReadKey = HKDF-SHA-512 of:

- InputKey = <Pair-verify shared secret>
- Salt = "Control-Salt"
- Info = "Control-Read-Encryption-Key"
- OutputSize = 32 bytes

ControlWriteKey = HKDF-SHA-512 of:

- InputKey = <Pair-verify shared secret>
- Salt = "Control-Salt"
- Info = "Control-Write-Encryption-Key"
- OutputSize = 32 bytes

The following key derivation scheme is used for the event connection:

EventReadKey = HKDF-SHA-512 of:

- InputKey = <Pair-verify shared secret>
- Salt = "Event-Salt"
- Info = "Event-Read-Encryption-Key"
- OutputSize = 32 bytes

EventWriteKey = HKDF-SHA-512 of:

- InputKey = <Pair-verify shared secret>
- Salt = ”Event-Salt”
- Info = ”Event-Write-Encryption-Key”
- OutputSize = 32 bytes

Note that the read/write keys are from the perspective of the device so their roles are reversed on the accessory (i.e. for the control connection accessory uses the write key for reading requests and the read key for writing responses; for the event connection it uses the read key for reading responses and the write key for writing requests). The AEAD algorithm is AEAD_CHACHA20_POLY1305 as specified in *ChaCha20-Poly1305*.

Each HTTP message is split into frames no larger than 16384 bytes. Each frame has the following format:

Table 3-29: Encrypted HTTP Frames

Bytes	Description
2	AAD for little endian length of encrypted data (n) in bytes
n	Encrypted data according to AEAD algorithm
16	authTag according to AEAD algorithm

3.3.2.2.2 Media Payload Encryption

All audio and video packets in CarPlay are encrypted using ChaCha20-Poly1305.

Each media stream must use a unique encryption key derived from the pair-verify shared secret, for each direction.

The following key derivation is used for the media streams. Note that the input/output keys are from the perspective of the device so their roles are reversed on the accessory.

DataStreamOutputKey = HKDF-SHA-512 of:

- InputKey = <Pair-verify shared secret>
- Salt = ”DataStream-Salt” + <stream connection ID>
- Info = ”DataStream-Output-Encryption-Key”
- OutputSize = 32 bytes

DataStreamInputKey = HKDF-SHA-512 of:

- InputKey = <Pair-verify shared secret>
- Salt = ”DataStream-Salt” + <stream connection ID>
- Info = ”DataStream-Input-Encryption-Key”
- OutputSize = 32 bytes

The stream connection IDs for individual media streams are delivered via RTSP SETUP command for those streams. See ”3.3.2.6 Streams” (page 120).

The encrypted payload is appended by the 16-bytes auth tag and potentially 8 bytes nonce for lossy protocols, such as UDP.

3.3.2.3 MFI-SAP

Authentication over the CarPlay interface requires use of the Apple Authentication Coprocessor for authenticating an encrypted channel. The protocol uses Curve25519 Elliptic-Curve Diffie-Hellman technology for key exchange, as described in `http://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 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 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 device.

When the 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.

3.3.2.4 Info Message

The info message provides device information to the accessory and returns accessory information to the device. It is an HTTP GET to the `/info` URL (see Table 3-77 (page 126)) with binary plist request and response payloads. The device may include any information it wants to provide to the accessory in the request plist. The request (see Table 3-30 (page 100)) may contain the "qualifier" key to limit the properties being requested from the accessory; for example, the device may only want to get the accessory's model. The response (see Table 3-32 (page 100)) contains the properties requested by the device.

Table 3-30: Info Message request keys

Key	Type	Required?	Description
qualifier	array	N	Array of property strings from Table 3-32 (page 100) 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.
uiContextURLs	array	N	Specifies an array of UI context URLs supported by the device. Only to be used for the "3.3.6 Shortcut Buttons" (page 155) feature. Accessory must be able to handle a dynamic array size (including 0). See Table 3-31 (page 100) for list of possible URLs.
altScreenURLs	array	N	Specifies an array of URLs that can be used with the showUI and suggestUI commands for instrument cluster displays. See "3.3.8.23 showUI" (page 174) for list of possible supported URLs. The accessory must be able to handle a dynamic array size (including 0).

Table 3-31: UI context URLs

Value	Description
"maps:"	Map or Navigation application
"mobilephone:"	Telephony application
"nowplaying:"	Media application
"messages:"	Messaging application

Table 3-32: 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 compatibility 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. Accessories supporting both CarPlay over USB and wireless CarPlay must always include all supported sample rates and codecs for each audioType for both transports. See Table 3-33 (page 105)
audioLatencies	array	Y	Array of dictionaries specifying the fixed audio latencies within the accessory. See Table 3-35 (page 107). 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).
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 "3.3.8.3 disableBluetooth" (page 163). * Required when accessory supports Bluetooth.
deviceID	string	Y	Globally unique ID for the accessory (e.g. the primary MAC address, such as "00:11:22:33:44:55"). Must match the value provided in DeviceIdentifier parameter of "15.6.2 CarPlayStartSession" (page 217).
displays	array	Y	An array of dictionaries for display information supported by the accessory. See Table 3-36 (page 107). The main display must always be the first element (index 0) in the array.
extendedFeatures	array	N*	Array of strings indicating support for additional features. See Table 3-42 (page 111). * Required when accessory supports CarPlay over wireless.

features	bitfield	Y	Internally defined for Apple CarPlay. Value must not be modified.
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 "3.3.5 User Input" (page 136).
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.
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 3-43 (page 112). * Required when limitedUI.
limitedUI	boolean	N	Indicates whether or not certain UI elements are limited. See "3.3.8.12 setLimitedUI" (page 170).
manufacturer	string	Y	Name of the accessory manufacturer.
model	string	Y	Model name of the accessory; must be globally unique and identify the acoustic environment of the product. The model name should include a unique identifier/code name of the vehicle brand, vehicle model, and a label for the unique acoustic tuning applied to the system.
modes	group	Y	The modes describing the current state of the accessory, including appState, mainScreen and mainAudio as applicable. See the Resource Management document on the MFi Portal for more details. See Table 3-44 (page 112).
name	string	Y	This must be set to "CarPlay".

nightMode	boolean	N*	Indicates if it is dark outside. The device can then use other metrics (such as time of day) to decide on its own when to enter night mode. See "3.3.8.11 setNightMode" (page 170). * The accessory must include this key if is capable of detecting whether it is night or day. The accessory must not include this key if it not capable of detecting whether it is night or day.
oemIcon	data	N*	PNG data for 104 x 104 pixel icon representing the accessory manufacturer's logo. The icon should be solid white with a transparent, alpha channel background. PNG data (as defined in ISO/IEC 15948:2004, see "3.1 Additional Specifications" (page 18)) must include IHDR, IDAT, IEND and sRGB chunks and optionally cHRM and gAMA, which are recommended. No other chunks can be included. There should be no shading or attempts to represent 3-dimensional volume since iOS will present the icon in the appropriate style (applying backgrounds, gradients, etc). The icon should follow the design principles of the current version of iOS. For example, see the Mail envelope icon. This icon will be used by devices prior to iOS 9.0. * Required when oemIconVisible is True.
oemIcons	array	N*	An array of dictionaries for icons representing the accessory manufacturer's logo. Icons which set the prerendered flag to True must be provided in the following sizes: 120x120, 180x180, 256x256. Icons which set the prerendered flag to False must be provided in the following sizes: 104x104, 156x156, 256x256. The device will use the image sized most appropriately for the usage and accessory screen size. See Table 3-46 (page 113). * 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	N	Protocol version string major.minor; e.g., "1.0". If present, the value must be "1.0".
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.
statusFlags	bitfield	Y	Status of an operation. Value must not be modified. See Table 3-47 (page 113).
vehicleInformation	group	N	Initial state of vehicle features. See Table 3-48 (page 114).
appearanceDefault	string	N*	Deprecated: Reporting appearanceDefault in the info message response has been replaced by reporting uiAppearanceMode in each display dictionary provided in the info message response. Indicates the accessory's UI theme setting when the device is connected. See Table 3-54 (page 115) for supported values. * Must only be included if accessory's UI theme setting is supported.
buttonInfo	array	Y	An array of dictionaries describing the location(s) and behavior of the vehicle's buttons to launch various applications in CarPlay. See Table 3-55 (page 115).
uiContextLastOnDisplayURLs	array	N	List of UI context URLs supported by the accessory shortcut buttons. URLs must be included in array provided in uiContextURLs of info message request Table 3-30 (page 100). See "3.3.6 Shortcut Buttons" (page 155).
uiContextNowOnDisplayURLs	array	N	List of UI context URLs to notify accessory when UI context is already on screen. URLs must be included in array provided in uiContextURLs of info message request Table 3-30 (page 100). See "3.3.8.10 requestUI" (page 168).
altScreenSuggestUIURLs	array	N	URLs that can be used with the "3.3.8.25 suggestUI" (page 176) command for instrument cluster displays. URLs must be included in array provided in altScreenURLs of info message request Table 3-30 (page 100).
enhancedSiriInfo	dictionary	Y	DEVELOPER PREVIEW Accessory's capabilities to activate Siri, see Table 3-61 (page 117).

Table 3-33: 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 3-72 (page 123)
audioInputFormats	bitfield	Y*	The audio input formats supported by the given type. See "3.2.7.2 Audio" (page 78) for requirements. * Do not include if the stream type does not support input. Combination of Table 3-34 (page 105).
audioOutputFormats	bitfield	Y	The audio output formats supported by the given type. See "3.2.7.2 Audio" (page 78) for requirements. Combination of Table 3-34 (page 105).
audioType	string	Y	The audio type to which the format(s) apply. See Table 3-73 (page 123)

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 types 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 device will use the same format for both directions and will choose from the supported input formats.

Table 3-34: Audio formats bitfield enum values

Value	Bit	Description
0x000000004	2	PCM, 8000 Hz, 16-Bit, Mono
0x000000008	3	PCM, 8000 Hz, 16-Bit, Stereo
0x000000010	4	PCM, 16000 Hz, 16-Bit, Mono
0x000000020	5	PCM, 16000 Hz, 16-Bit, Stereo
0x000000040	6	PCM, 24000 Hz, 16-Bit, Mono
0x000000080	7	PCM, 24000 Hz, 16-Bit, Stereo
0x000000100	8	PCM, 32000 Hz, 16-Bit, Mono
0x000000200	9	PCM, 32000 Hz, 16-Bit, Stereo
0x000000400	10	PCM, 44100 Hz, 16-Bit, Mono
0x000000800	11	PCM, 44100 Hz, 16-Bit, Stereo
0x000001000	12	Reserved
0x000002000	13	Reserved

0x000004000	14	PCM, 48000 Hz, 16-Bit, Mono
0x000008000	15	PCM, 48000 Hz, 16-Bit, Stereo
0x000010000	16	Reserved
0x000020000	17	Reserved
0x000040000	18	Reserved
0x000080000	19	Reserved
0x000100000	20	Reserved
0x000200000	21	Reserved
0x000400000	22	AAC-LC, 44100 Hz, Stereo
0x000800000	23	AAC-LC, 48000 Hz, Stereo
0x002000000	25	DEVELOPER PREVIEW Deprecated: AAC-ELD, 48000 Hz, Stereo
0x004000000	26	DEVELOPER PREVIEW Deprecated: AAC-ELD, 16000 Hz, Mono
0x008000000	27	DEVELOPER PREVIEW Deprecated: AAC-ELD, 24000 Hz, Mono
0x010000000	28	OPUS, 16000 Hz, Mono
0x020000000	29	OPUS, 24000 Hz, Mono
0x040000000	30	OPUS, 48000 Hz, Mono
0x080000000	31	DEVELOPER PREVIEW Deprecated: AAC-ELD, 44100 Hz, Mono
0x100000000	32	DEVELOPER PREVIEW Deprecated: AAC-ELD, 48000 Hz, Mono
0x800000000	43	DEVELOPER PREVIEW Deprecated: AAC-ELD, 32000 Hz, Mono

DEVELOPER PREVIEW

Deprecated: Use of AAC-ELD has been deprecated.

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 OPUS compression must be used for the low latency "main audio" streams (except for "entertainment" audio) and alternate audio. See "3.2.7.2 Audio" (page 78).

Table 3-35: 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 3-72 (page 123).
audioType	string	Y	Type of audio content (e.g. telephony, media, etc.). See Table 3-73 (page 123).
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 3-36: Display keys

Key	Type	Required?	Description
features	bitfield	Y	Specifies features of the display as a bitmask. Combination of Table 3-40 (page 110).
maxFPS	number	N	Max frames per second the display supports.
heightPhysical	number	Y	Specifies the height of the display in millimeters. Must be non-zero.
widthPhysical	number	Y	Specifies the width of the display in millimeters. Must be non-zero.
heightPixels	number	Y	Specifies the height of the display in pixels. Must be non-zero.
widthPixels	number	Y	Specifies the width of the display in pixels. Must be non-zero.
uuid	string	Y	Specifies the UUID of the display.
primaryInputDevice	enum	Y	Specifies the primary input device to be used for navigating the user interface. One of Table 3-41 (page 111).

viewAreas	array	N*	Specifies an array of dictionaries for view area information supported by the accessory. See Table 3-37 (page 109). * Required if "viewAreas" included in enabledFeatures array in Setup response, see Table 3-65 (page 119).
initialViewArea	number	N*	Specifies the index of the initial view area. * Required if "viewAreas" array is included.
adjacentViewAreas	array	N*	Specifies an array of viewAreas indexes that describe the view areas that can be transitioned from the view area identified by initialViewArea. Must contain exactly one element if the view area identified by initialViewArea has viewAreaTransitionControl set to True. * Required if "viewAreas" array is included.
cornerMasks	boolean	N	Indicates the accessory will apply corner clipping masks on this display. Must not be included for instrument cluster displays. See "3.2.7.1.7 Corner Clipping Masks" (page 73).
initialURL	string	N	URL of UI content to be shown on alternate display at connection. URL must be included in array provided in altScreenURLs of info message request Table 3-30 (page 100). This must not be included for the main display.
type	enum	Y	Specifies if the display is the main display or an instrument cluster display surface. Uses same values as specified in Table 3-72 (page 123).
accessoryGiveFocus	boolean	N	Specifies if the device owns focus on connection. Must only be included when the accessory supports "3.3.5.4 UI Focus Transfer" (page 150). Must not be included for instrument cluster displays.
uiAppearanceMode	enum	Y	Specifies the accessory's system-wide UI appearance mode for the display. One of Table 3-59 (page 116). See "3.2.7.1.9 Appearance Modes" (page 77).
uiAppearanceSetting	enum	Y	Specifies the accessory's system-wide UI appearance setting for the display. One of Table 3-60 (page 116). See "3.2.7.1.9 Appearance Modes" (page 77).
mapAppearanceMode	enum	N*	Specifies the accessory's map appearance mode for the display. One of Table 3-59 (page 116). * See "3.2.7.1.9 Appearance Modes" (page 77).
mapAppearanceSetting	enum	N*	Specifies the accessory's map appearance setting for the display. One of Table 3-60 (page 116). * See "3.2.7.1.9 Appearance Modes" (page 77).

Table 3-37 lists the viewArea keys.

Table 3-37: viewArea keys

Key	Type	Required?	Description
widthPixels	number	Y	Specifies the width in pixels. Must be a multiple of 2.
heightPixels	number	Y	Specifies the height in pixels. Must be a multiple of 2.
originXPixels	number	Y	Specifies the X coordinate in pixels. The top left corner is at coordinates (0,0). Must be a multiple of 2.
originYPixels	number	Y	Specifies the Y coordinate in pixels. The top left corner is at coordinates (0,0). Must be a multiple of 2.
safeArea	group	N	Specifies the size and location of the safe area. Must not be included for the main display if corner clipping masks are supported. See "3.2.7.1.7 Corner Clipping Masks" (page 73)
viewAreaTransitionControl	boolean	Y*	If True shows a control in the CarPlay UI to transition to another view area. * Main display only.
viewAreaSupportsFocusTransfer	boolean	Y*	If True the view area represents a UI configuration where focus transfer is supported. * Main display only.
viewAreaStatusBarEdge	enum	Y*	Specifies the position of the status bar when the view area is active. One of Table 3-39 (page 110). See "3.2.7.1.8 Status Bar Position" (page 77). Consult with Apple for usage guidelines. * Main display only.

Table 3-38: safeArea keys

Key	Type	Required?	Description
widthPixels	number	Y	Specifies the width in pixels. Must be a multiple of 2.
heightPixels	number	Y	Specifies the height in pixels. Must be a multiple of 2.
originXPixels	number	Y	Specifies the X coordinate in pixels. The top left corner is at coordinates (0,0). Must be a multiple of 2.
originYPixels	number	Y	Specifies the Y coordinate in pixels. The top left corner is at coordinates (0,0). Must be a multiple of 2.
drawUIOutsideSafeArea	boolean	N	DEVELOPER PREVIEW Specifies whether CarPlay UI is drawn outside of the safe area boundary on the main display. Consult with Apple for usage guidelines. See "3.2.7.1.5 Safe Area" (page 68). Main display only.

Table 3-39: viewAreaStatusBarEdge enum values

Value	Description
0x00	Status bar position will be determined by the device.
0x01	Status bar will be positioned at the bottom of the CarPlay UI.
0x02	Status bar will be positioned on the driver side of the CarPlay UI.

Table 3-40: 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 3-41: 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 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 touchscreen 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 3-42 shows the string values for CarPlay extended features. If specified, the accessory supports the enhanced feature.

Table 3-42: Extended features string values

Value	Description
enhancedRequestCarUI	Indicates accessory support for the `url` request key of the `requestUI` command. See "3.3.8.10 requestUI" (page 168).
vocoderInfo	If included the vocoderInfo parameter will be included in audio stream setup message requests (see Table 3-67 (page 120)).

Table 3-43: 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 3-44: Initial Mode keys

Key	Type	Required?	Description
appStates	array	Y	Application state(s). Contains dictionaries of Table 3-98 (page 164).
resources	array	Y	Resource ownership(s). Contains dictionaries of Table 3-99 (page 165).
initialPermanentEntity	array	Y	Identifies the initial permanent owner of each resource. Must always contain a dictionary for each resource. Contains dictionaries of Table 3-45 (page 112).

Table 3-45: initialPermanentEntity keys

Key	Type	Required?	Description
permanentEntity	enum	Y	Identifies the permanent owner of the resource. Value must not be 'None'. One of Table 3-81 (page 131).
resourceID	enum	Y	Identifies the resource. One of Table 3-82 (page 131).
takeConstraint	enum	N*	One of Table 3-85 (page 133). * Required if permanentEntity is accessory.
borrowConstraint	enum	N*	One of Table 3-85 (page 133). * Required if permanentEntity is accessory.

Table 3-46: Icon keys

Key	Type	Required?	Description
imageData	data	Y	PNG data for an icon - PNG data (as defined in ISO/IEC 15948:2004, see "3.1 Additional Specifications" (page 18)) must include IHDR, IDAT, IEND and sRGB chunks and optionally cHRM and gAMA, which are recommended. No other chunks can be included. 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. Icon corners must be square, the 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	Width in pixels of the image.
prerendered	boolean	Y	Indicates if the icon is full-bleed. False if the icon is a mask icon.

Table 3-47: 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 `/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).

Table 3-48: Vehicle Information keys

Key	Type	Required?	Description
ElectronicTollCollection	group	N	If this key is present, the accessory supports Electronic Toll Collection (Japan), and the ETC status symbol will be shown within the CarPlay UI. The Intelligent Transport Systems Technology Enhancement Association (ITS-TEA) may require a logo license to use this parameter; see `https://www.its-tea.or.jp`. See Table 3-49 (page 114)
NavigationAidedDriving	group	N	If this key is present, the accessory supports Navigation Aided Driving, i.e. the vehicle uses active route guidance from the native navigation system as an input for driving assistance systems. Destination Information, see "5 Destination Information" (page 181), must be supported if this key is present. See Table 3-50 (page 114)
userPreferences	group	N	A group of dynamic and user defined preferences. See Table 3-51 (page 114).

Table 3-49: Electronic Toll Collection keys

Key	Type	Required?	Description
active	boolean	Y	Whether or not Electronic Toll Collection (Japan) is currently active in the vehicle.

Table 3-50: Navigation Aided Driving keys

Key	Type	Required?	Description
active	boolean	Y	Whether or not the vehicle is currently using active route guidance in the native navigation system as an input for the vehicle's driving assistance systems.

Table 3-51: User Preferences keys

Key	Type	Required?	Description
touchpadSettings	group	Y*	A group of touchpad specific settings for each identified HID touchpad, see Table 3-88 (page 137), keyed by the respective HID uuid. See Table 3-52 (page 115). * Required if accessory has a touchpad.

Table 3-52: Touchpad Settings keys

Key	Type	Required?	Description
touchpadSensitivity	number	N	Sensitivity of the touchpad device from 0.0 (slowest allowed sensitivity) to 1.0 (fastest allowed sensitivity). 0.5 represents the default sensitivity.
supportedHapticFeedbackTypes	bitfield	Y*	Haptic feedback types supported by the touchpad, see Table 3-53 (page 115). If not supplied, no haptic feedback will be performed. * Required if accessory touchpad supports haptic feedback.

Table 3-53: Haptic Feedback Types bitfield enum values

Value	Bit	Description
0x1	0	Default. If the accessory supports multiple haptic feedback types, this represents the most common type used by the accessory.

Deprecated: Reporting appearanceDefault in the info message response has been replaced by reporting uiAppearanceMode in each display dictionary provided in the info message response.

Table 3-54: `appearanceDefault` string values

Value	Description
"automatic"	Accessory has both light and dark UI themes and switches between them automatically based on if it is light or dark outside.

Table 3-55: Button Information keys

Key	Type	Required?	Description
buttonType	enum	Y	The type of application that this button launches. See Table 3-56 (page 116).
buttonLocation	enum	Y	The button's location in the vehicle. See Table 3-57 (page 116).
buttonPressDuration	enum	Y	The button's press duration to launch the corresponding application in CarPlay. See Table 3-58 (page 116).

Table 3-56: Button Type enum values

Value	Description
0x00	Siri.

Table 3-57: Button Location enum values

Value	Description
0x00	Steering wheel.
0x01	Center console.
0x02	Not located on steering wheel or center console.

Table 3-58: Button Press Duration enum values

Value	Description
0x00	Short press.
0x01	Long press.

Table 3-59: Appearance Mode enum values

Value	Description
0x00	Light.
0x01	Dark.

Table 3-60: Appearance Setting enum values

Value	Description
0x00	Automatic.
0x01	User choice.
0x02	Always.

DEVELOPER PREVIEW

Table 3-61: enhancedSiriInfo keys

Key	Type	Required?	Description
enhancedSiriVoice	boolean	Y	Indicates support for launching Siri by voice, see "3.3.7.2 Voice Activation from the Accessory" (page 159).
enhancedSiriButton	boolean	Y	Indicates support for instant activation on button press, see "3.3.7.1 Instant Button Activation from the Accessory" (page 157).
voiceModelSupportedLanguages	array	Y(*)	Array of languages supported by the voice trigger model in the accessory. The accessory must be able to switch to a supported language upon a request by the device. Represented as strings, see "3.3.7.2.1 Keyword Detection Mode" (page 160). (*) Required if enhancedSiriVoice is True.
voiceModelCurrentLanguage	string	Y(*)	Currently set language of the voice model, see "3.3.7.2.1 Keyword Detection Mode" (page 160). (*) Required if enhancedSiriVoice is True.
supportedSiriTriggerZones	bitfield	Y(*)	Bitmask describing the location in the vehicle the keyword was detected. (*) Required if enhancedSiriVoice is True. Table 3-62 (page 117).

DEVELOPER PREVIEW

Table 3-62: supportedSiriTriggerZones bitfield enum values

Value	Bit	Description
0x00000001	1	Driver

3.3.2.5 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 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 3-63 on the following page lists the specifics of the initial setup request keys sent by the device to the accessory.

Table 3-63: Initial setup request keys

Key	Type	Required?	Description
deviceID	string	Y	Globally unique device ID (e.g. ”11:22:33:44:55”).
macAddress	string	Y	MAC address of the iOS network interface used for the connection.
model	string	Y	Model name of the device (e.g. ”Device1,1”).
name	string	Y	User-customizable name of the device.
osBuildVersion	string	Y	Operating system build version string (e.g. ”11A200”).
sessionUUID	string	Y	UUID of the session.
sourceVersion	string	Y	Device side CarPlay source version string (e.g. ”101.7”).
streams	array	Y*	Specifies an array of stream descriptions to set up and configure the audio and screen streams. See Table 3-67 (page 120) and Table 3-70 (page 122). * The initial setup message may not include `streams` if it only sets up the control stream.
timingPort	number	Y	Port the device is listening on for time sync requests.
features	array	N	Specifies features supported by the device. To learn more, see Table 3-64 (page 118).
type	enum	N	Specifies the stream type. To learn more, see Table 3-72 (page 123).

When establishing a CarPlay session, the device may advertise support for optional features through the `features` array.

Table 3-64: `features` string values

Value	Description
”viewAreas”	Indicates that the device supports view areas. See ”3.2.7.1.3 Display Information” (page 65)
”uiContext”	Indicates that the device supports UI context. See ”3.3.6 Shortcut Buttons” (page 155)
”cornerMasks”	Indicates that the device supports corner clipping masks. See ”3.2.7.1.7 Corner Clipping Masks” (page 73)
”focusTransfer”	Indicates that the device supports focus transfer. See ”3.3.5.4 UI Focus Transfer” (page 150)
”altScreen”	Indicates that the device supports rendering CarPlay content to an instrument cluster. See ”3.2.7.1.2 Instrument Cluster Display” (page 64)
”enhancedSiri”	DEVELOPER PREVIEW Indicates that the device supports launching Siri by voice, instant activation on button press and auxiliary audio streams. See ”3.3.7 Activating Siri” (page 157).

Table 3-65 lists the specifics of the initial setup response keys sent by the accessory to the device.

Table 3-65: Initial Setup response keys

Key	Type	Required?	Description
eventPort	number	Y	TCP/UDP port number for events.
enabledFeatures	array	Y*	DEVELOPER PREVIEW Specifies features supported by the accessory. See Table 3-66 (page 119). * Required if features array in initial setup request includes 'enhancedSiri'.
keepAlivePort	number	Y	UDP port number for session idle UDP keepalive.
streams	array	Y*	Array of stream descriptions. See Table 3-67 (page 120) and Table 3-70 (page 122). * The initial setup message may not include `streams` if it only sets up the control stream.
timingPort	number	Y	Port the device is listening on for time sync requests.

The `enabledFeatures` array enables the accessory to identify the features it supports.

Table 3-66: `enabledFeatures` string values

Value	Description
"viewAreas"	If included, the accessory supports view areas. See "3.2.7.1.3 Display Information" (page 65)
"uiContext"	If included, the accessory supports UI context. See "3.3.6 Shortcut Buttons" (page 155)
"cornerMasks"	If included, the accessory supports corner clipping masks. See "3.2.7.1.7 Corner Clipping Masks" (page 73)
"focusTransfer"	If included, the accessory supports focus transfer. See "3.3.5.4 UI Focus Transfer" (page 150)
"altScreen"	If included, the accessory supports rendering CarPlay content to an instrument cluster. See "3.2.7.1.2 Instrument Cluster Display" (page 64)
"enhancedSiri"	DEVELOPER PREVIEW Accessory supports instant Siri activation on button press and auxiliary audio streams. See "3.3.7 Activating Siri" (page 157).

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.

3.3.2.6 Streams

Each stream is set up using a stream descriptor with a type, parameters, and behaviors associated with it, as listed in Table 3-67 (page 120) for main/alternate audio stream and Table 3-70 (page 122) 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 "3.3.2.9 Synchronization" (page 127).

Main/alternate audio streams and the screen stream for the main display are only set up when the device is ready to send audio or video content to the accessory, and are torn down when no longer required. If supported, screen streams for instrument cluster displays are set up immediately after the CarPlay session is established and are not torn down for the remainder the session.

Table 3-67: 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 3-34 (page 105).
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 accommodated 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. The accessory must not modify the value provided by the device.
audioType	string	Y*	Type of audio content (e.g. telephony, media, etc.). See Table 3-73 (page 123). * Main audio only.
controlPort	number	Y*	UDP port the device is listening on for RTCP requests. * Main High audio only.
dataPort	number	Y*	UDP port the 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).
spf	number	Y*	Number of audio samples in a frame. Note: stereo samples are counted as a single sample (e.g., 352 stereo samples would not mean 704 samples per frame). * Main High audio only.

streamConnectionID	number	Y	Unique stream connection ID. Used for deriving encryption keys.
type	enum	Y	Type of stream. One of Table 3-72 (page 123).
vocoderInfo	group	Y	Telephony vocoder information. Contains dictionary of Table 3-68 (page 121).
supportsRTPPacketRedundancy	boolean	Y*	Device supports redundant audio data. *Only required if device supports redundant audio data.
streamStartTimestamp	number	Y*	NTP-synchronized time value starting from which audio data should be streamed from the circular buffer. * Auxiliary input audio only.
burstPeriodMs	number	Y*	Time period for bursting audio data, in milliseconds. Audio should be accumulated every burstPeriodMs time interval. This value may override any previously sent burstPeriodMs values. This value must not be changed by the accessory. * Auxiliary input audio only.

Table 3-68: Vocoder Information keys

Key	Type	Required?	Description
`sampleRate`	number	Y	The native sample rate of the audio data in the telephony output stream. For cellular calls this describes the sample rate a call is encoded across the cellular network.

Table 3-69: Audio stream descriptors response keys

Key	Type	Required?	Description
controlPort	number	Y*	UDP port the accessory is listening on for RTCP requests. * Main High audio only.
dataPort	number	Y	UDP port the accessory is listening on for data.
type	enum	Y	Type of stream. One of Table 3-72 (page 123)
sampleTime	number	N	Media timestamp at the same moment as "timestamp".
streamConnectionID	number	Y	Unique stream connection ID.
timestamp	number	N	Wall clock timestamp at the same moment as "sampleTime" using synchronized wall time.
timestampRawNs	number	N	Unadjusted wall clock timestamp.
supportsRTPPacketRedundancy	boolean	Y*	Accessory supports redundant audio data. *Only required if setup request includes supportsRTPPacketRedundancy. See Table 3-67 (page 120).

Table 3-70: 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 3-72 (page 123)
streamConnectionID	number	Y	Unique stream connection ID. Used for deriving encryption keys.
topLeftCornerMask	data	N	Corner clipping mask grayscale PNG data (as defined in ISO/IEC 15948:2004, see "3.1 Additional Specifications" (page 18)). Included if accessory supports "3.2.7.1.7 Corner Clipping Masks" (page 73).
uuid	string	Y	Specifies the UUID of the display.

Table 3-71: 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 3-72 (page 123)

Table 3-72 on the following page shows the ID values of streams sent from the device to the accessory:

Table 3-72: 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.
Aux Out Audio	106	UDP	DEVELOPER PREVIEW Low-latency speech output. Value is also the RTP payload type.
Aux In Audio	107	UDP	DEVELOPER PREVIEW Low-latency speech input. Value is also the RTP payload type.
Screen	110	TCP	H.264 screen output for the main display.
Alt Screen	111	TCP	H.264 screen output for an instrument cluster display.

Table 3-73 shows string values for CarPlay audio types:

Table 3-73: 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, Aux Out Audio, Aux In Audio	DEVELOPER PREVIEW Input and output for speech recognition and Siri.
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 3-74: Channels

Name	Transport	Port	QoS	Direction	Description
RTSP Controller control	TCP	5000	BE	device to accessory	iOS control
RTSP Accessory control	TCP	dynamic	BE	accessory to device	Accessory control
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 High Audio Output	UDP	dynamic	VO	device to accessory	Audio data
RTCP Main High Audio Output	UDP	dynamic	VO	device to accessory	Audio control
RTP Main Audio Input	UDP	dynamic	VO	accessory to device	Audio data
RTP Auxiliary Audio Input	UDP	dynamic	VO	accessory to device	DEVELOPER PREVIEW Audio data
RTP Auxiliary Audio Output	UDP	dynamic	VO	device to accessory	DEVELOPER PREVIEW Audio data
Time Sync Client	UDP	dynamic	VO	accessory to device	Audio time sync requests
Time Sync Server	UDP	dynamic	VO	device to accessory	Audio time sync responses
RTP Screen	TCP	dynamic	VI	device to accessory	User interface (H.264 frames)
Keep Alive	UDP	dynamic	BK	device to accessory	Keep alive request

3.3.2.7 Feedback Message

The feedback message exchanges timing information and statistics to and from an accessory. The 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 "3.3.2.9 Synchronization" (page 127).

Table 3-75: Feedback response keys

Key	Type	Required	Description
`streams`	array	N	Streams reporting feedback. Contains dictionaries of Table 3-76 (page 125)

Table 3-76: 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 3-72 (page 123).
sampleTime	number	Y	Media timestamp at the same moment as "timestamp".
streamConnectionID	number	Y	Unique stream connection ID.
timestamp	number	Y	Wall clock timestamp at the same moment as "sampleTime" using synchronized wall time.
timestampRawNs	number	Y	Unadjusted wall clock timestamp.

3.3.2.8 URLs

Table 3-77 (page 126) lists the URLs used with CarPlay. All URLs are accessed via HTTP/RTSP.

Table 3-77: URLs for CarPlay

URL	Method	Description
n/a	FLUSH	RTSP message to flush any queued audio. Main High audio only.
n/a	OPTIONS	Standard HTTP options message to return the supported methods, etc.
n/a	RECORD	RTSP message to start playback.
n/a	SET_PARAMETER	DEVELOPER PREVIEW RTSP message to setup a set of parameters. See Table 3-78 (page 126).
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 "3.3.2.3 MFI-SAP" (page 99). Request and response are opaque security data.
`/command`	POST	For performing command requests and their responses; see "3.3.8 Commands" (page 162). Requests and responses are binary plists.
`/feedback`	POST	Exchanges information with the accessory; see "3.3.2.7 Feedback Message" (page 124). Requests and responses are binary plists.
`/info`	GET	Exchanges information with the accessory; see "3.3.2.4 Info Message" (page 99). Requests and responses are binary plists.
`/logs`	GET	Gets logs from the accessory. Response is a gzip archive of log data.
`/pair-setup`	POST	Perform pairing setup; see "3.3.2.1 Pairing" (page 97).
`/pair-verify`	POST	Perform pairing verification; see "3.3.2.1 Pairing" (page 97).

DEVELOPER PREVIEW

Table 3-78: SET_PARAMETER method keys

Key	Type	Required?	Description
enhancedSiriParameters	group	Y	Setup parameters for the enhancedSiri feature. Contains a dictionary of siriParameters keys. Only sent if both accessory and device support the enhancedSiri feature, see Table 3-79 (page 127).

Table 3-79: enhancedSiriParameters keys

Key	Type	Required?	Description
voiceActivationMode	enum	Y	Sets the mode of the voice detector. See Table 3-80 (page 127).
bufferAudioFormat	bitfield	Y	Bitfield indicating the audio format to be used to buffer audio into the circular buffer. Either OPUS, 16000 Hz, Mono for wireless CarPlay or PCM, 16000 Hz, 16-Bit, Mono for CarPlay over USB. See Table 3-34 (page 105).
bufferSizeMs	number	Y	In milliseconds, the size of the voice buffer. This parameter will not exceed 5000ms. The bufferSizeMs will be provided by the device at the beginning of a session and will not change during an active CarPlay session. The value can change between sessions.
voiceModelLanguage	string	N*	The language setting to be used for Keyword Detection mode. "3.3.7.2.1 Keyword Detection Mode" (page 160). Present only when voiceActivationMode is set to VoiceKeyword. * The parameter will not be sent if voiceActivationMode is set to voiceActivity or disabled.
burstPeriodMs	number	N*	Time period for bursting audio data from the circular buffer to the device, in milliseconds. Audio should be accumulated every burstPeriodMs time interval. This value may override any previously sent burstPeriodMs values. This value must not be changed by the accessory. * Auxiliary input audio only.

Table 3-80: voiceActivationMode values

Name	Value	Description
disabled	-1	When this mode is set, the accessory must stop sending any voice activation events.
voiceKeyword	1	When this mode is set, the accessory must switch to using the Keyword Detection Mode.
voiceActivity	2	When this mode is set, the accessory must switch to using the Voice Activity Detection Mode.

3.3.2.9 Synchronization

The clocks between the 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 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 device.

3.3.2.9.1 Wall Clock Synchronization

Synchronizing wall clocks is done by exchanging Network Time Protocol (NTP) packets. The 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.

3.3.2.9.2 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 device. This allows the 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 device to determine the absolute media clock offset between devices. Media clock timing relationships are exchanged via the "3.3.2.7 Feedback Message" (page 124).

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).

3.3.2.10 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.

3.3.3 Resource Management

3.3.3.1 Resources

DEVELOPER PREVIEW

In its typical embodiment in an automobile, the CarPlay architecture has two shared resources:

- **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 device) for main audio. Speech recognition, telephony, and media 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 "3.3.3.3 Resource Ownership" (page 130)).

Auxiliary and alternate audio streams do not compete for access to the audio resource. Auxiliary input and output audio may be active in the background during audio playback and must not interrupt media playback. Auxiliary input audio is set up by the device without any update to resources.

When the device activates Siri, the device will update appState(speech) and may claim the Screen resource. When Siri ends, the device will update appState(speech) and may release the Screen resource. While Siri is active Main Audio remains unchanged and stays with the current owner since media playback continues in the background.

Similarly, when the accessory sets up native voice recognition, it must update the appState(speech) and may borrow the Screen resource. When native voice recognition ends, the accessory must update the appState(speech) and may unborrow the Screen resource. The accessory may choose either to mix native recognition prompts with music playback, in which case the owner of Main Audio remains unchanged, or interrupt media playback by borrowing the Main Audio resource.

3rd party apps will continue to use Main Audio "speechRecognition" and will continue to use both Screen and Main Audio resources.

An instrument cluster display does not participate in resource management in the same way as the main display. See "3.2.7.1.2 Instrument Cluster Display" (page 64) for information on displaying CarPlay UI content on an instrument cluster display.

3.3.3.2 App States

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

- **Speech**: An entity (accessory or 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 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.

3.3.3.3 Resource Ownership

Only one entity (accessory or 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 direct user 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. The accessory may initiate a direct user action on the user's behalf only 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 a direct user action 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 device's Music app, for example, the 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 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 current owner (entity) may change temporarily, however the other entity will retain permanent ownership of the resource (permanent entity) 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 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 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.

A resource must not be taken with a constraint of Never. Instead, the accessory should borrow the resource (with an unborrow constraint of Never) and then unborrow the resource when finished with it.

3.3.3.4 Examples of Resource Management

For examples of mode changes, see the *Resource Management* document on the MFi Portal.

3.3.4 Modes

Modes provide a mechanism to manage shared resources (e.g. the main screen), and application state (e.g. on a phone call) during a CarPlay session. The accessory must provide a set of initial modes during connection establishment, see "3.3.4.1 Initial Mode" (page 133). During an active session, the accessory may request access to these shared resources, see "3.3.8.4 changeModes" (page 164). Any time a resource owner temporarily or permanently changes, it is announced to the accessory. To learn more, see "3.3.8.5 modesChanged" (page 165).

The entities involved in resource management are as follows:

Table 3-81: 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 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.

The resources being managed are as follows:

Table 3-82: 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 device or the accessory may use it but not both at once.

For each resource, a transfer type must be specified. The transfer types are as follows:

Table 3-83: Resource ownership transfer type enum values

Name	Value	Description
Take	1	Acquire ownership of a resource permanently. For example, if the user switches to the accessory's FM radio from the device's music app, the accessory would request to "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 an entity requests it.
Borrow	3	Transfer ownership temporarily. For example, if the user is listening to the device's music app and the accessory starts a native voice recognition, the accessory would "borrow" main audio. When the native voice recognition session ends, the accessory device would "unborrow" main audio.
Unborrow	4	Release ownership of a resource that was acquired temporarily. Ownership of a resource returns to the previous owner.

For transfer types take and borrow, a transfer priority is required to indicate the reason for the transfer. The priorities are as follows:

Table 3-84: 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.

There are multiple types of constraints for each transfer type:

- takeConstraint sets the minimum priority a subsequent take request must have to successfully take the resource.
- borrowConstraint sets the minimum priority a subsequent borrow request must have to successfully borrow the resource.
- unborrowConstraint sets the minimum priority a subsequent unborrow request must have to successfully unborrow the resource.

The following constraints are required for each transfer type:

- Take: `takeConstraint` and `borrowConstraint`
- Borrow: `unborrowConstraint`
- Untake: no constraints
- Unborrow: no constraints

The constraint values are as follows:

Table 3-85: 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.

If the accessory is engaged in one of the activities in the following table, it must indicate this to the device. The device will also provide its app state to the accessory.

Table 3-86: 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.

When the app state is speech, the state is further specified by values in the following table:

Table 3-87: Speech mode enum values

Name	Value	Description
None	-1	No speech-related states are active.
Speaking	1	An entity is speaking to the user. This includes in-progress phone calls after the ring tone has finished.
Recognizing Speech	2	An entity is recording audio to recognize speech from the user.

3.3.4.1 Initial Mode

When a session starts (on connection, for example), the device queries the accessory for its current mode to synchronize the states of the two devices. This is done via the "3.3.2.4 Info Message" (page 99). The accessory must provide information about the desired current owner, including the type, priority and constraints for each resource, along with the permanent owner including the entity and constraints for each resource, as well as current app states and speech mode using the modes dictionary. The 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 device prior to connection to the accessory, the device would then "take" main audio, provided that this action was compatible with the accessory's initial mode constraints.

If the accessory wants to claim permanent ownership (take) for a resource at connection time, the accessory must provide an initialPermanentEntity dictionary per resource and include accessory as the permanentEntity along with the resourceID, takeConstraint, and borrowConstraint. The accessory must provide the same takeConstraint and borrowConstraint in both the resources dictionary as well as the initialPermanentEntity dictionary (see Table 3-45 (page 112)).

If the accessory wants to claim temporary ownership (borrow) for a resource at connection time, the accessory must use the resources dictionary to define the borrow, and at same time it must declare a desired permanent owner for when a unborrow is sent. Take requests received after the initial modes may overwrite the initial permanent owner declared in the info response message.

If the accessory wants to transfer permanent ownership for a resource to iOS at connection time, the accessory must provide a initialPermanentEntity dictionary per resource and include controller as the permanentEntity along with the resourceID. The accessory must not include takeConstraint and borrowConstraint in this case.

Accessory must continue to populate the resources dictionary as previously specified without any changes for backwards compatibility with older iOS versions.

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.

3.3.4.2 Mode Changes

To modify the current mode, the accessory sends the device a mode change request communicating the intent of the mode change. This is done via the `/command` URL (See "3.3.8 Commands" (page 162)). The device takes the request and determine a new mode based on it. The 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 device has evaluated a new mode, it communicates it to the accessory (see "3.3.4.3 Current Mode" (page 136)). Mode changes originating from the 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 device to the accessory.

The accessory must honor any updates to the current mode within 100 ms of receiving them from the device. For example, it must relinquish control of the screen within 100 ms if screen ownership is transferred to the 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, device may Take back Main Screen: Anytime or User Initiated or Never
  - For Transfer Type Take, device may Borrow back Main Screen: Anytime or User Initiated or Never
  - For Transfer Type Borrow, device may Unborrow Main Screen: Anytime or User Initiated or Never
- Borrow Identifier, for Transfer Type Borrow and Unborrow: identifies the borrow which is removed by an unborrow.

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, device may Take back Main Audio: Anytime or User Initiated or Never
  - For Transfer Type Take, device may Borrow back Main Audio: Anytime or User Initiated or Never
  - For Transfer Type Borrow, device may Unborrow Main Audio: Anytime or User Initiated or Never
- Borrow Identifier, for Transfer Type Borrow and Unborrow: identifies the borrow which is removed by an unborrow.

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 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 "3.3.3.3 Resource Ownership" (page 130). 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 unless the resource was transferred to the other entity, for example, if an accessory borrows main audio 3 times, it must unborrow 3 times to release it to the previous owner. However if the accessory has borrowed the resource and then the controller became the owner, the accessory does not have to specifically unborrow the resource. The release transfer type indicates to the device that the accessory no longer needs the resource, and the resource constraints will be reset to "Other may take anytime". A successful taking of a resource currently borrowed by the other entity will terminate the borrow, as described in "3.3.3.3 Resource Ownership" (page 130).

The accessory can use a priority value to indicate to the 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." A priority must not be set for untake and unborrow.

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 device within 200 ms of the time that the accessory issues the `changeModes` command (see "3.3.8.4 changeModes" (page 164)) or receives a modesChanged command (see "3.3.8.5 modesChanged" (page 165)).

3.3.4.3 Current Mode

The current mode is communicated by the device using the modesChanged command, see "3.3.8.5 modesChanged" (page 165). The command includes the following parameters (see Table 3-103 (page 166)):

- **Accessory** or **Device** has Screen currently (the result of an entity borrowing or taking Screen)
- **Accessory** or **Device** has Screen permanently (the result of an entity taking Screen)
- **Accessory** or **Device** has Main Audio currently (the result of an entity borrowing or taking Main Audio)
- **Accessory** or **Device** has Main Audio permanently (the result of an entity taking Main Audio)
- **Accessory** or **Device** or **Neither** is performing speech recognition or speech
- **Accessory** or **Device** or **Neither** is engaged in a phone call
- **Accessory** or **Device** or **Neither** is performing turn-by-turn navigation

The accessory must monitor the modesChanged (see "3.3.8.5 modesChanged" (page 165)) command continuously. If ownership of a resource is returned to the accessory when the accessory didn't explicitly request it, the accessory must resume its previous activity for that resource.

3.3.5 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, see `http://www.usb.org/developers/hidpage/`.

The accessory must comply with the following requirements:

- User events must be represented as Human Interface Device (HID) usage reports and must be transported using the CarPlay communication protocol.
- Accessories must not use iAP2 to transport HID events while using CarPlay, with one exception: accessories supporting HID Media Playback Remote may send those commands (except for Voice Control / Siri) over iAP2, *HID Media Playback Remote* as defined in the *Accessory Interface Specification*.
- When supporting CarPlay, the Voice Command / Siri usage must always be transported using the CarPlay communication protocol. To send HID events over CarPlay use the command requestSiri as described in "3.3.8.9 requestSiri" (page 167).
- Usage of map zoom level controls for an instrument cluster display must be transported using the CarPlay communication protocol as described in "3.2.7.1.2 Instrument Cluster Display" (page 64). DEVELOPER PREVIEW

See additional requirements in *HID* as defined in the *Accessory Interface Specification*.

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 3-88 (page 137) lists the possible HID device description keys stored in an HID device dictionary.

Table 3-88: 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.
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	ID to uniquely identify the HID device within this CarPlay session.

3.3.5.1 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:

- Touchscreen
- Touchpad

3.3.5.1.1 Touchscreen

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

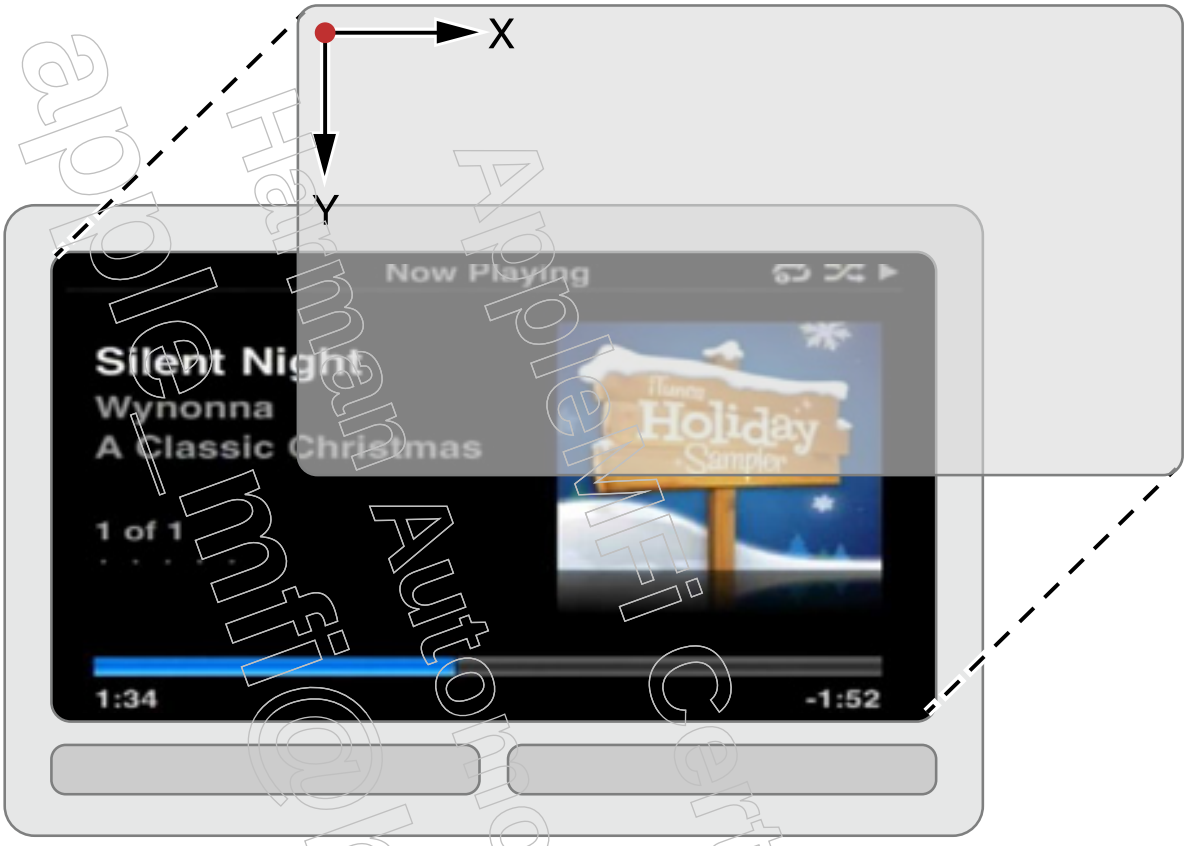

Figure 3-21: Touchscreen

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 "3.2.2.1 High Resolution Displays" (page 20) and "3.2.7.1 User Interface Streams" (page 62).

3.3.5.1.2 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 3-22: Touchpad

Generally, reports originating from this class of device would entail 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 "3.3.5.1.4 Single Touch" (page 140)
- Button 1 (Primary Button) to convey selection, see "3.3.5.1.5 Buttons Accompanying Single Touch" (page 141)
- AC Back button, see "3.3.5.1.5 Buttons Accompanying Single Touch" (page 141)

If the accessory supports character recognition, it must provide character input gesture support (see "3.3.5.2 Character Input Gesture Support" (page 142)) using hidSetInputMode (see "3.3.8.8 hidSetInputMode" (page 167)).

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

See "3.3.8.8 hidSetInputMode" (page 167) to configure the input mode of a HID device and "3.3.8.7 hidSendReport" (page 166) for delivering HID reports.

3.3.5.1.3 HID Descriptor Support

The accessory may support the following HID usages:

Table 3-89: 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
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 device does not own the screen, the accessory must not send these HID usages.

3.3.5.1.4 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 for a Touch Pad:

- Physical Minimum
- Physical Maximum
- Unit Exponent
- Unit

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

```
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)
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
```

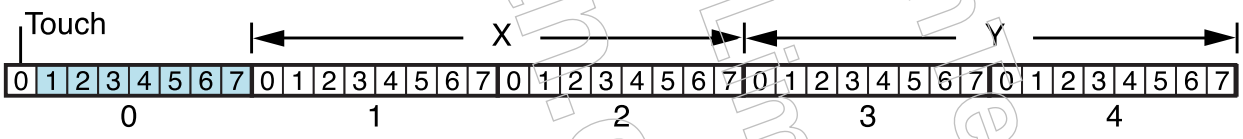

Figure 3-23: Example Input Report Layout for Single-Touch Touchscreen

3.3.5.1.5 Buttons Accompanying Single Touch

In order to convey selection and the back button, an accessory must declare an application collection with the following usages:

- Button 1 (Primary Button) to convey selection, see "3.3.5.1.5 Buttons Accompanying Single Touch" (page 141)

- AC Back button, see "3.3.5.1.5 Buttons Accompanying Single Touch" (page 141)

The following is an example report descriptor and format for the buttons accompanying a single-touch touchscreen:

```
0x05, 0x0C,         // Usage Page (Consumer)
0x09, 0x01,         // Usage (Consumer Control)
0xA1, 0x01,         // Collection (Application)
0x05, 0x09,         // Usage Page (Button)
0x09, 0x01,         // Usage (Button 1 primary/trigger)
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, 0x24, 0x02,   // Usage (AC Back)
0x95, 0x01,         // Report Count (1)
0x81, 0x02,         // Input (Data, Variable, Absolute)
0x95, 0x06,         // Report Size (6)
0x81, 0x01,         // Input (Constant)
0xC0                //End Collection
```

3.3.5.2 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 3-90: 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
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 device does not own the screen, the accessory must not send these HID usages.

3.3.5.2.1 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 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 device will notify the accessory of the input mode using the hidSetInputMode as described in "3.3.8.8 hidSetInputMode" (page 167).

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)
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
```

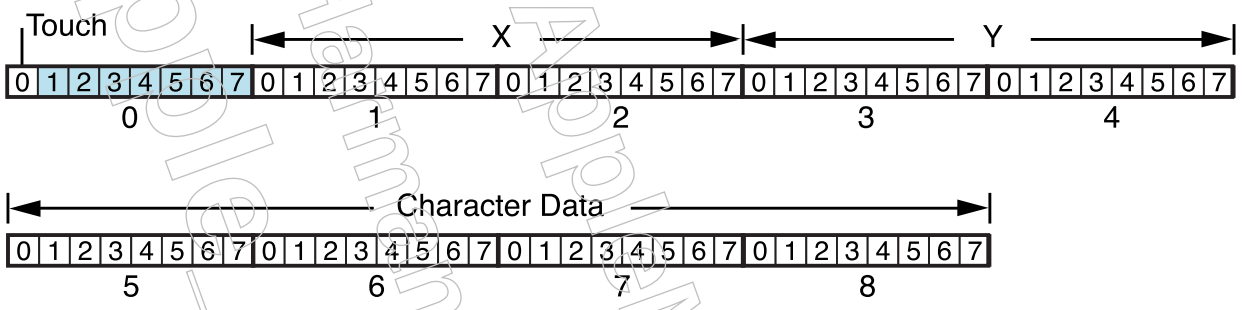

Figure 3-24: Example Input Report Layout for Single-Touch Touchpad with Basic Character Gesture Recognition

When reporting character data, care must be 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
```

3.3.5.2.2 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 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 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 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
```

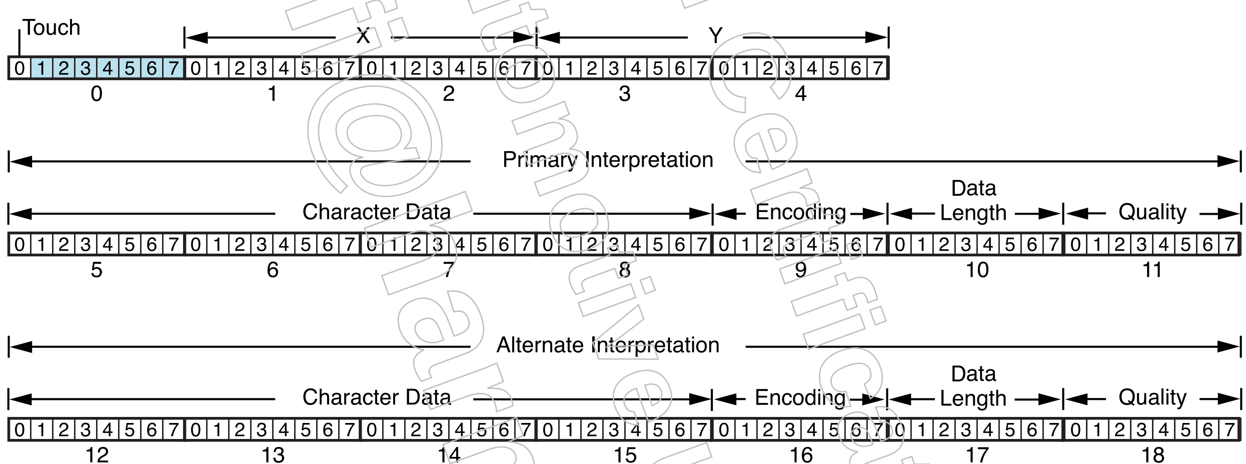

Figure 3-25: Example Input Report Layout for Single-Touch Touchpad with Character Gesture Recognition with Alternate Interpretations

3.3.5.3 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 minimally consists of variable and rotational axis Z, and it may also consist of variable axes X and Y.

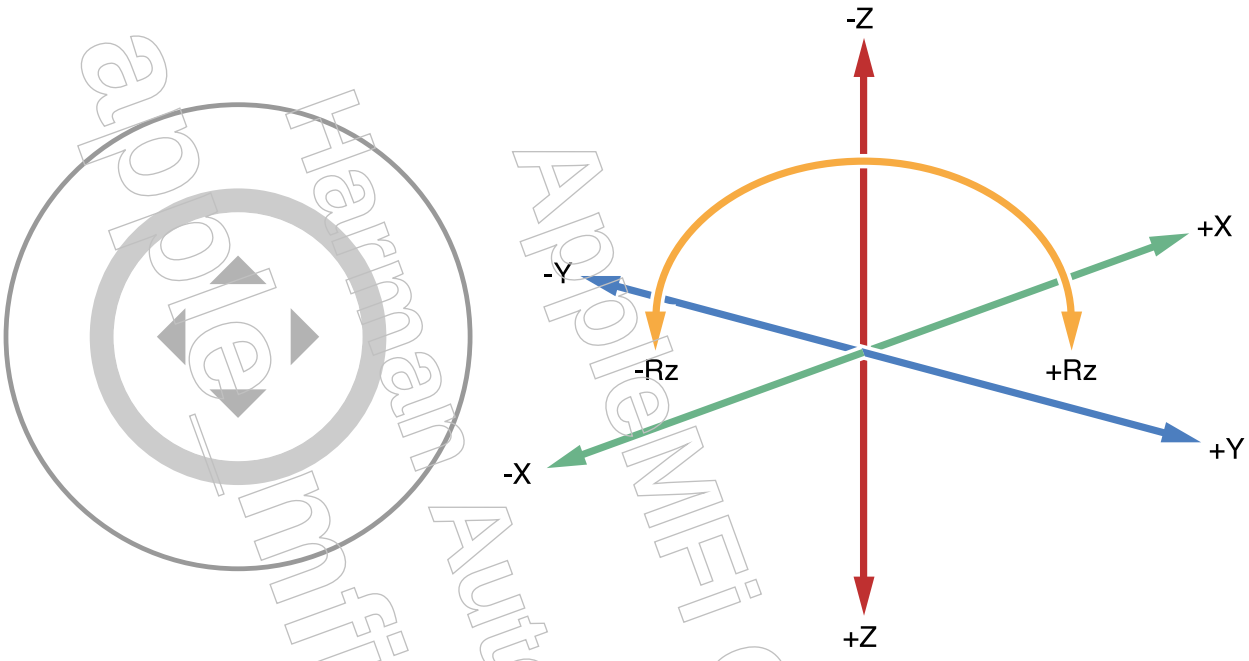

Figure 3-26: Multi-axis Controller

The accessory may support the following HID usages:

Table 3-91: 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)
0x0C	Consumer	0x224	AC Back	One Shot Control

When the 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 either one of the following to convey selection:

- Z
- Button 1 (Primary Button)

When implementing the Z axis, the device only requires movement in the +Z (down) direction and will translate 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 if the accessory uses a knob as the primary input. It is recommended that all accessories with a physical Back button support AC Back.

If the accessory has translation movement, it must support:

- X
- Y

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
```

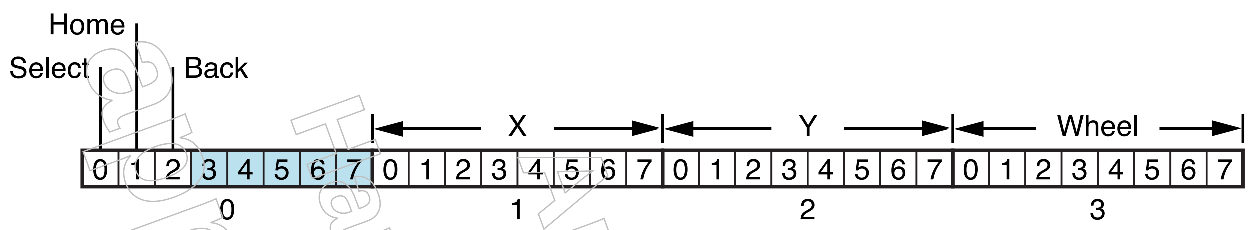

Figure 3-27: Example Input Report Layout for Multi-Axis Controller

3.3.5.4 UI Focus Transfer

Accessories that use knob or touchpad user input devices typically show a visual highlight when a UI element is reached by the user through a rotation of a knob or a finger swipe on a touchpad. This visual highlight indicates that the UI element currently has focus and if the user presses down on the knob or the touchpad the UI element will be selected.

When CarPlay occupies the full display, focus is managed within the CarPlay UI. However, if the CarPlay UI is shown in a windowed configuration and is adjacent to native UI elements that can be focused, the UI focus needs to be transferred between CarPlay and the native system as the user navigates the UI.

Accessories with the following configuration must support UI focus transfer:

- Knob and/or touchpad user input devices
- CarPlay is shown in a windowed configuration adjacent to focusable native UI elements

The accessory must negotiate support for UI focus transfer using the initial setup message and info message. The device declares support by including a 'focusTransfer' string in the features array within the initial setup message request (see Table 3-64 (page 118)).

If the device supports UI focus transfer, the accessory must include a 'focusTransfer' string in the enabledFeatures array within the initial setup message response (see Table 3-66 (page 119)). The accessory must also include the viewAreaSupportsFocusTransfer parameter in the view area dictionary representing any UI configuration where CarPlay UI is adjacent to native UI elements that can be focused (see Table 3-37 (page 109)).

If the device does not support UI focus transfer, the accessory must not include a 'focusTransfer' string in the enabledFeatures array within the initial setup message response and must not include the viewAreaSupportsFocusTransfer parameter in any view area dictionaries.

When focus is transferred the following information about the currently focused UI element may be provided by the entity that focus will be transferred from:

- X coordinate for the top left corner, relative to the display
- Y coordinate for the top left corner, relative to the display
- Width in pixels
- Height in pixels
- The heading from which focus is being transferred

3.3.5.4.1 Transferring Focus from Native UI to CarPlay UI

When a native UI element is focused and the CarPlay UI is also shown on the display the accessory must evaluate whether focus should be passed to CarPlay whenever a knob or touchpad user input event occurs. When this evaluation takes place, the CarPlay UI should be considered as any other control would within the native UI focus navigation concept.

When focus needs to be passed to CarPlay the accessory must:

1. Remove any highlight treatment from the focused UI element
2. Send a "3.3.8.20 accessoryGiveFocus" (page 173) command
3. Send knob or touchpad user input events as HID reports to CarPlay

If the device does not support UI focus transfer the accessory must not send any "3.3.8.20 accessoryGiveFocus" (page 173) commands.

3.3.5.4.2 Transferring Focus from CarPlay UI to Native UI

When knob or touchpad HID reports are sent to CarPlay focus will move around the UI elements within the CarPlay UI. If the user reaches the last focusable UI element within the CarPlay UI and continues moving the knob or touchpad in the same direction CarPlay will send a deviceOfferFocus "3.3.8.21 deviceOfferFocus" (page 173) command.

If there is a native UI element adjacent to the CarPlay UI in a position that would be the logical place to transfer focus to, the accessory must:

1. Send a "3.3.8.22 accessoryAcquireFocus" (page 174)) command in response to a "3.3.8.21 deviceOfferFocus" (page 173) command
2. Stop sending knob or touchpad user input events as HID reports to CarPlay
3. Apply highlight treatment to the native UI element that is now focused

If there are no native UI elements adjacent to the CarPlay UI in a position that would be the logical place to transfer focus to, the accessory must not respond to "3.3.8.21 deviceOfferFocus" (page 173) commands.

3.3.5.4.3 Initial Focus

DEVELOPER PREVIEW

When a CarPlay session starts the accessory must determine if focus is transferred to CarPlay, using accessoryGiveFocus in the main display dictionary (see Table 3-36 (page 107)) in the info message response (see "3.3.2.4 Info Message" (page 99)):

- If the CarPlay UI replaces a native UI element that has focus, the accessory must transfer focus to CarPlay by setting the value of accessoryGiveFocus to True.
- If a native UI element has focus, and is still visible when the CarPlay UI is shown, the accessory must not transfer focus to CarPlay. In this case the value of accessoryGiveFocus must be set to False.

3.3.5.5 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 3-92: Button Support HID Usages

Page ID	Page Name	Usage ID	Usage Name	Usage Type	Apple Function
0x0C	Consumer	0xB0	Play	On/Off Control	Play
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
0x0C	Consumer	0x29E	AC Navigation Guidance	One Shot Control	Play the last navigation guidance prompt
0x07	Keyboard	0x2A	Keyboard Delete (Backspace)	One Shot Control	Clear
0x0B	Telephony	0x20	Hook Switch	On/Off Control	Accept, Hold or Toggle call
0x0B	Telephony	0x21	Flash	Momentary Control	Toggle Accept or Reject/End call
0x0B	Telephony	0x26	Drop	One Shot Control	Reject/End call or End Siri session
0x0B	Telephony	0x2F	Phone Mute	On/Off 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
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 physical telephony buttons must implement the following:

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

Telephony HID Commands must only be used for physical buttons. Telephony controls shown on a display must use Call Controls feature via iAP2, a sub-feature of Communications (see *Accessory Interface Specification*).

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
```

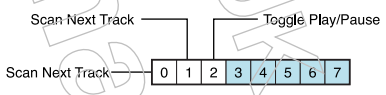

Figure 3-28: Example Input Report Layout for Simple Media Buttons

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
```

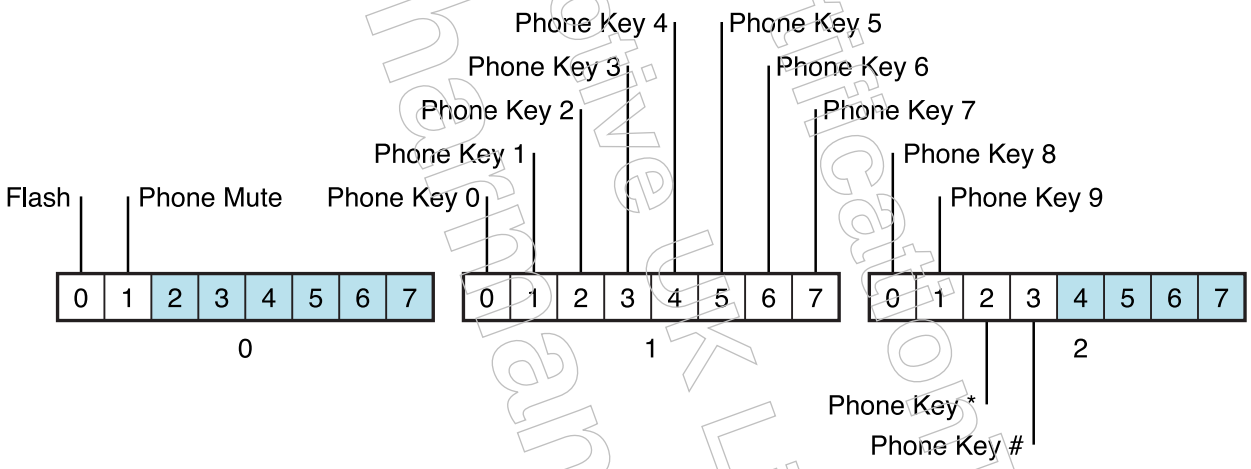

Figure 3-29: Example Input Report Layout for Simple Telephone Buttons

3.3.5.6 Human Presence

A CarPlay accessory may report human presence to indicate when the user may interact with the accessory.

The accessory may support the following HID usages:

Table 3-93: Human Presence Support HID Usages

Page ID	Page Name	Usage ID	Usage Name	Usage Type	Apple Function
0x20	Sensor	0x11	Biometric: Human Presence	Application Collection	Sensor Present
0x20	Sensor	0x04B1	Data Field: Biometric Human Presence	Dynamic Flag	Human Present

These usages must be added as a logical collection to other top level collections to indicate the physical location of the sensor. For example, if the sensor is located near a touchscreen, a sensor logical collection should be added to the digitizer HID descriptor.

The accessory must only report Human Present as 1 when it can reliably detect biometric presence at a distance where interaction with a control is intended or about to occur.

The following is an example report descriptor and format for human presence:

```
0x05, 0x20,          // Usage Page (Sensor)
0x09, 0x11,          // Usage 0x11 (Biometric Human Presence)
0xA1, 0x01,          // Collection (Application)
0x05, 0x20,          //   Usage Page (Sensor)
0x0A, 0xB1, 0x04,    //   Usage 0x4B1 (Sensor Data Biometric Human Presence)
0x15, 0x00,          //   Logical Minimum......... (0)
0x25, 0x01,          //   Logical Maximum......... (1)
0x75, 0x08,          //   Report Size............. (8)
0x95, 0x01,          //   Report Count............ (1)
0x81, 0x02,          //   Input................... (Data, Variable, Absolute)
0xC0,                // End Collection
```

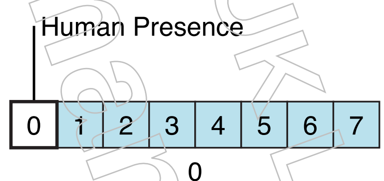

Figure 3-30: Example Input Report Layout for Human Presence

3.3.6 Shortcut Buttons

If the accessory has physical buttons, or virtual buttons on a touchscreen that provide direct access to application categories such as:

- Telephony
- Navigation and/or Maps

it must support shortcut buttons in CarPlay as described in *CarPlay Design Guidelines*.

For shortcut button behaviour that requires the accessory to know if an application category was last shown in the CarPlay UI or native UI the accessory must use "3.3.6.1 UI Context" (page 156).

3.3.6.1 UI Context

UI context describes if an application category was last shown on the display in the CarPlay UI or native UI.

The accessory must declare support for UI context in the SETUP Response message (see "3.3.2.5 Setup Message" (page 117)) and declare only the application categories needed to satisfy the behaviors described in *CarPlay Design Guidelines* using uiContextLastOnDisplayURLs parameter in the info message response (see "3.3.2.4 Info Message" (page 99)).

Both accessory and device send "3.3.8.19 changeUIContext" (page 172) when the UI context changes for an application category. If an application category is displayed in the CarPlay UI the device evaluates if the same application category was displayed in native UI since it was last shown in the CarPlay UI. "3.3.8.19 changeUIContext" (page 172) will only be sent by the device if the application category was last displayed in native UI.

Similarly, the accessory may evaluate if an application category was last displayed in the CarPlay UI before sending "3.3.8.19 changeUIContext" (page 172). If the application category was last shown in the CarPlay UI the accessory must send "3.3.8.19 changeUIContext" (page 172). Alternatively the accessory may send "3.3.8.19 changeUIContext" (page 172) whenever an application category is shown in native UI, without evaluating if that application category was last displayed in the CarPlay UI.

If an application category is not included in uiContextURLs in the info request message (see "3.3.2.4 Info Message" (page 99)) the UI context of that application category is not required be updated by the accessory.

The accessory must not send the "3.3.8.19 changeUIContext" (page 172) message for partial UI elements such as status indicators, widgets or secondary display windows. For example, if a map widget that occupies a secondary area of the main display is shown to the user a "3.3.8.19 changeUIContext" (page 172) must not be sent.

3.3.6.2 Telephony

Accessories supporting shortcut buttons for telephony application categories must be used to show CarPlay telephony UI using the `requestUI(mobilephone:)` command. Refer to *CarPlay Design Guidelines* for details on shortcut buttons labelled 'PHONE' or 'TEL'.

3.3.6.3 Navigation and Map

Accessories supporting shortcut buttons for navigation and/or maps application categories must show a navigation or map UI either in native UI or in the CarPlay UI. Refer to *CarPlay Design Guidelines* for details on shortcut buttons labelled 'NAVI' or 'MAP'.

The vehicle may be fitted with a single shortcut button that opens a combined map and navigation feature, or two different shortcut buttons: a dedicated Map button and a dedicated Navigation button.

- If the vehicle console is fitted with a single Map or Navigation shortcut button: the `requestUI(maps:)` command must be used to show maps/navigation UI in CarPlay.

- If the vehicle console is fitted with separate Map and Navigation shortcut buttons:
  - The `requestUI(maps:)` command must be used with the Map button to show map UI in CarPlay.
  - The `requestUI(maps:/car/destinations)` command must be used with the Navigation button to show navigation UI in CarPlay.

3.3.6.4 Alternate Actions

Some shortcut buttons may change behavior based on the UI being shown when the button is pressed. For example, a navigation shortcut button may alternate between showing a map view or a destination entry view, depending on whether a map view is already visible on the screen. If the accessory implements such behavior, it must implement the same behavior for equivalent CarPlay application categories by considering what is currently visible in the CarPlay UI.

To implement such alternate actions, the accessory must declare the CarPlay application categories required for this behavior using the uiContextNowOnDisplayURLs parameter in the info message response, see "3.3.2.4 Info Message" (page 99).

When the user presses a shortcut button with possible alternate actions, the accessory must send a `requestUI` command to the device with a URL indicating the appropriate application category. If CarPlay is already showing the requested application category, the device will respond with a `requestUI` command using the same URL. This is an indication that the alternate action should occur. The accessory must react to the received `requestUI` command and show an alternate UI to match the native behavior.

3.3.7 Activating Siri

DEVELOPER PREVIEW

When Siri is activated from the accessory, the accessory must send a requestSiri command which must include an activation type (siriAction) and an activation timestamp (siriTriggerTimestamp). See "3.3.8.9 requestSiri" (page 167). Based on the provided timestamp and the activation type, the device will set up an auxiliary in audio stream and request a subset of the audio stored in the circular buffer to be streamed starting at streamStartTimestamp, see Table 3-67 (page 120). streamStartTimestamp may match the time when Siri was activated or it may be before the activation event. However, it will never exceed the size of the circular buffer which is set by the device at the beginning of each CarPlay session (bufferSizeMs, see Table 3-79 (page 127)). streamStartTimestamp is provided each time the auxiliary in audio stream is set up, see Table 3-67 (page 120).

When the audio stream is set up, relevant pre-buffered audio is transferred faster than real-time to the device. Once audio catches up to the current time, streaming continues in real-time until the audio input stream is closed by the device. Such handling allows for both one-shot Siri interactions where the user says the full request in one go (e.g. "Hey Siri, get directions to the closest gas station."), as well as two-shot interactions where the user first initiates Siri by saying "Hey Siri" and then speaks the request (e.g. "Get directions to the closest gas station").

Accessories must report the location and behavior of buttons in the vehicle that can activate Siri, see "3.3.2.4 Info Message" (page 99).

3.3.7.1 Instant Button Activation from the Accessory

The accessory must support instant activation for all buttons that can be used to activate Siri. The Siri button can be dedicated to Siri, or overloaded to support other native features.

3.3.7.1.1 Overloaded Speech Recognition Button

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 value for the longer press timeout must not be greater than 600 ms. It is important that the accessory prewarm Siri as soon as the button is physically pressed to ensure the best possible response time from Siri in the event that it is activated. After the initial 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.

For each requestSiri command—Prewarm, buttonDown, ButtonUp—the accessory must include the time of the physical button presses as well as the the time of expiration of the long press timeout as (siriTriggerTimestamp). The requestSiri command must be sent within 50 ms of the user interaction with the button.

Short press (accessory speech recognition):

```
Physical button depressed
|
|        Physical button released
|        |
t1-------t2-------------------------------------------------------------------------> time
|        |
|        requestSiri (siriAction:ButtonUp,timestamp:t2)
|
requestSiri (siriAction:Prewarm,timestamp:t1)
```

For short presses, where the physical button is released before the accessory-determined timeout interval, the Prewarm action is sent to the device when the button is pressed, the ButtonUp action is sent when the button is released, and the accessory's native speech recognition system is activated. In this case, no requestSiri commands should be sent until the accessory's voice recognition session completes.

Long press (Siri):

```
Physical button depressed
|
|                Long Press Timeout      Physical button released
|                |                       |
t1---------------t2----------------------t3--------------------------------------> time
|                |                       |
|                |                       requestSiri (siriAction:ButtonUp,timestamp:t3)
|                |
|                requestSiri (siriAction:ButtonDown,timestamp:t2)
|
requestSiri (siriAction:Prewarm,timestamp:t1)
```

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 device along with the timestamps of when those event occured, 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.

3.3.7.1.2 Dedicated Siri Button

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. For each button press, the accessory must include the time of the button press (siriTriggerTimestamp).

The requestSiri command must be sent within 50 ms of the user interaction with the button.

Dedicated button (Siri):

```
Physical button depressed
|
|               Physical button released
|               |
t1--------------t2-------------------------------------------------------------> time
|               |
|               requestSiri (siriAction:ButtonUp,timestamp:t2)
|
requestSiri (siriAction:ButtonDown,timestamp:t1)
```

Note: The accessory's voice recognition system is expected to end when the user triggers a Siri interaction.

3.3.7.2 Voice Activation from the Accessory

To support launching Siri by voice, accessories must support three voice detection modes:

- Keyword Detection mode.
- Voice Activity Detection (VAD) mode.
- Deactivated.

The device configures the mode of operation when a CarPlay session is established and may change the mode at any time during an active session using the SET_PARAMETER method, see Table 3-78 SET_PARAMETER method keys (page 126). At the beginning of each CarPlay session, or at any time during a session, the accessory must be able to update its detection mode based on the value of voiceActivationMode in the SET_PARAMETER method. Using the same method, the device may also disable voice activation.

Regardless of which detection mode is used, the device uses a second pass voice trigger module to further analyze the speech utterance. The second pass voice trigger module makes the final determination to launch Siri and interact with the user.

While the second pass voice trigger module is processing, Siri is not active and the accessory must remain in its current state. There must not be any changes in media playback.

If the second pass voice trigger module determines that the user requested Siri (e.g. "Hey Siri" was said), Siri will be launched and shown in CarPlay. The device will update appState (speech) and may gain access to the Screen resource by sending a modesChanged command (see "3.3.3 Resource Management" (page 128)). While Siri is active, the device may send a duckAudio command to indicate that the volume of the currently playing audio must be reduced.

If the second pass does not confirm that the user requested Siri, auxiliary input audio will be torn down and the accessory must resume analyzing the microphone audio and send requestSiri(voiceActivation) commands when applicable. The accessory must not send any requestSiri(voiceActivation) commands while appState(speech) is set.

The accessory must request voice activation only when the user is actively speaking. For example, a podcast, a radio talk show played through the car speakers, or non-spoken sounds in the cabin must not be detected as voice activations. The accessory must continue to detect and request Siri activations while any main audio type except telephony or speech recognition, and/or alternate audio is active in CarPlay.

3.3.7.2.1 Keyword Detection Mode

Keyword Detection mode, also known as voice trigger detection, is a technique used to detect the presence of a specific word in speech. In this mode, the accessory must analyze all speech activity and detect any utterance of "Siri" in its uplink signal. Once the accessory has detected the keyword "Siri," it must send a requestSiri(voiceActivation) command and provide a siriTriggerTimestamp representing when the start of the keyword was spoken (not when it was finished). For example, the siriTriggerTimestamp must be the time at which the 'S' was detected and not the 'i' in "Siri". In Keyword Detection mode, the accessory must:

- When the device sets up an auxiliary input audio stream, start streaming audio to the device within 50 ms.
- Analyze and detect in real time the keyword "Siri".
- The latency from receiving the end of the "Siri" keyword (second "i" sound) at the accessory's microphone until keyword detection is completed must not exceed 420 ms.
- Determine when the start of the keyword ("S") was spoken and provide it in siriTriggerTimestamp.
- Report a keyword voice trigger event to the device by sending a requestSiri command with the parameter voiceActivation and siriTriggerTimestamp within 50 ms from when the word was detected.
- False Reject Rate (FRR) must not exceed 1% for a vaid Siri request.
- False Acceptance Rate (FAR) must not exceed 2 per hour of active external speech audio.

Voice Activation in Keyword Detection Mode:

```
      'Hey Siri whats the time'
       |   |
------t1--t2---------------------------------------------------------------------> time
               |
               requestSiri (siriAction:VoiceActivation,timestamp:t2)
```

The accessory must advertise the currently active language and a list of languages that are supported in keyword detection mode in the info message response. Languages are specified using the format [language designator]-[region designator], where the language designator uses ISO 639-1, and the region designator uses ISO 3166-1. For more details, see `https://developer.apple.com/library/archive/documentation/MacOSX/Conceptual/BPInternational/LanguageandLocaleIDs/LanguageandLocaleIDs.html`.

If the language of the accessory's keyword detector model does not match with the current Siri language, the device will send a SET_PARAMETER message to update the value of voiceModelLanguage, see Table 3-79 enhancedSiriParameters keys (page 127). When the message is received, the accessory must switch the language of the keyword detection mode to the language provided in voiceModelLanguage.

3.3.7.2.2 Voice Activity Detection

Voice Activity Detection mode, also known as speech detection, is a technique used in signal processing to detect the presence and absence of speech. In this mode, the accessory must detect a human engaged in talking, or a speech activity. The accessory must detect when there is the presence of speech after a period of absence of speech. This transition is known as a Start-of-Speech (SoS) event, and the accessory must keep a timestamp of this event. If 300 ms of continuous speech follows, the accessory must send a requestSiri(voiceActivation) command and provide siriTriggerTimestamp of when the SoS event was detected. The detection algorithm should classify uplink signal into two categories—speech and silence (non-speech). To maintain high speech recognition and task completion accuracy, SoS events should be identified accurately and communicated to the device in a timely manner, along with the expected buffered audio. In Voice Activity Detection mode, the accessory must:

- When the device sets up the auxiliary input audio stream, stream audio to the device in less than 50 ms.
- Detect a speech region when there is continuous speech activity present for at least 300 ms. If there is a non-speech region shorter than 400 ms between two speech regions of at least 300 ms each, treat them as one speech region.
- Sounds from the accessory's speakers (radio, music, navigation prompts, voice-interaction system) and ambient sounds inside or around the car (turn signals, horns, traffic, environmental) must not trigger SoS events.
- The latency from receiving a speech region at the accessory's microphone until Start Of Speech (SoS) is detected must not exceed 140 ms.
- Determine the start of the speech region and provide it in siriTriggerTimestamp.
- Report a SoS event to the device by sending a requestSiri command with the parameter voiceActivation and siriTriggerTimestamp within 50 ms from when when a spoken word was was detected.
- Speech Miss Rate (SMR) - rate of missing speech activity must not exceed 0.1% per hour for a typical conversation.
- False Wake Rate (FWR) - false speech activity reports must not exceed 15 wakes per hour.

Voice Activation in Voice Detection Mode:

```
       'Hey Siri whats the time'
        |   |
-------t1--t2-------------------------------------------------------------------> time
          |
          requestSiri (siriAction:VoiceActivation,timestamp:t1)
```

3.3.7.2.3 Deactivated

In this state the accessory must disable voice activation for Siri and must not send any requestSiri(voiceActivation) commands.

3.3.7.3 Activating from the Device

When Siri is activated from within the CarPlay user interface, or on the device, the device will set up the auxiliary input audio stream and request a subset of buffered audio to be streamed starting at streamStartTimestamp. The accessory must respond to such requests in the same way as when Siri is activated from the accessory.

3.3.7.4 Multiple Voice Assistants

Accessories may support multiple voice-activated assistants. Voice activation for all assistants, including Siri, must be available concurrently when there is no active assistant. Before a specific assistant is launched, all assistants may evaluate the voice input to determine if the request was directed at them. Once an assistant has determined that the user requested activation, the assistant can launch. When an assistant is launched, the accessory must update appState(speech), and may borrow the Screen resource if needed. Once the assistant completes its task, the accessory must update appState(speech), and may unborrow the Screen resource if appropriate.

While an assistant is active, no other assistant can be activated by voice. The accessory must not forward the user's audio input to another voice assistant while there is an on-going conversation with the active assistant, and the accessory must not send any requestSiri(voiceActivation) commands to the device. If the active assistant also handles physical button presses, any buttons occurring during the active session should be forwarded to the active assistant.

For example, if the accessory supports both voice-activated Siri launch and voice-activated launch of a native assistant, the following rules apply:

- When there is no active assistant, both Siri and the native assistant may evaluate the user's speech to determine if the request is targeted to them. At this point, none of the assistants is showing any UI to the user.
- If the user asked for the native assistant, the accessory signals to the device that the native assistant is starting using appState(speech) and if applicable, by borrowing the Screen. Then the Controller stops listening for voice activations, and the accessory must stop sending requestSiri(voiceActivation) commands—Siri cannot interrupt the native assistant by voice activation.
- If the user asked for Siri, the controller signals that Siri is starting using appState(speech). The accessory must stop listening for other voice activations and cannot interrupt Siri by voice activation.
- For physical buttons that are shared between Siri and another assistant, if Siri is active, all button presses must be sent to the device.
- If Siri is active, and if there is a dedicated button on the accessory which always activates another assistant, the accessory may interrupt Siri and launch the other assistant.
- If a native assistant is active, and there is a dedicated button on the accessory which always activates Siri, the accessory must interrupt the native assistant and request Siri to start.
- When the native assistant ends, the accessory must update appState(speech) and may unborrow the Screen.
- When a Siri session ends, the controller will update the appState and may unborrow the Screen.

3.3.8 Commands

Commands are messages exchanged between the device and the accessory to perform an action. They are sent as HTTP requests using the `/command` URL on the control stream. The 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>
```

3.3.8.1 duckAudio

Sender: Device

Description: Ramps volume down (for example, to fade down music to play a navigation prompt). See "3.2.7.2.13 Ducking" (page 91). Table 3-94 (page 163) lists the specifics of duckAudio request keys.

Table 3-94: duckAudio request keys

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

3.3.8.2 unduckAudio

Sender: Device

Description: Ramps volume back to the pre-duck level. See "3.2.7.2.13 Ducking" (page 91). Table 3-95 (page 163) lists the specifics of unduckAudio request keys.

Table 3-95: unduckAudio request keys

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

3.3.8.3 disableBluetooth

Sender: Device

Description: Disable Bluetooth connectivity to specified device (i.e. the iOS device). If the accessory supports Bluetooth, it must provide its bluetoothIDs (see "3.3.2.4 Info Message" (page 99)) 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 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 device while the two are already connected over CarPlay (that is, a session is already in progress). Table 3-96 (page 164) lists the specifics of disableBluetooth request keys.

Table 3-96: disableBluetooth request keys

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

3.3.8.4 changeModes

Sender: Accessory

Description: Change resource states and/or app states. See "3.3.4 Modes" (page 131). Table 3-97 (page 164) lists the specifics of changeModes request keys and Table 3-100 (page 165) lists the specifics of changeModes response keys.

Table 3-97: changeModes request keys

Key	Type	Required?	Description
appStates	array	N	Application state(s) to change. Contains dictionaries of Table 3-98 (page 164).
resources	array	N	Resource ownership(s) to change. Contains dictionaries of Table 3-99 (page 165).
reasonStr	string	N	A textual description of the reason for the mode change.

Table 3-98: appState keys

Key	Type	Required?	Description
appStateID	enum	Y	One of Table 3-86 (page 133).
speechMode	enum	N*	One of Table 3-87 (page 133). * Required if `appStateID` is Speech.
state	boolean	N*	* Required if `appStateID` is `PhoneCall` or `TurnByTurn`.

Table 3-99: resource keys

Key	Type	Required?	Description
resourceID	enum	Y	Identifies the resource. One of Table 3-82 (page 131).
transferType	enum	Y	One of Table 3-83 (page 132).
transferPriority	enum	N*	One of Table 3-84 (page 132). * Required if `transferType` is take or borrow.
takeConstraint	enum	N*	One of Table 3-85 (page 133). * Required if `transferType` is take.
borrowConstraint	enum	N*	One of Table 3-85 (page 133). * Required if `transferType` is take.
unborrowConstraint	enum	N*	One of Table 3-85 (page 133). * Required if `transferType` is borrow.
borrowID	string	N*	String identifier to match each borrow and unborrow request. * Required if the `transferType` is borrow or unborrow.

Table 3-100: `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.

3.3.8.5 modesChanged

Sender: Device

Description: Updates the accessory's current mode after a mode change by the device. Table 3-101 (page 165) lists the specifics of modesChanged request keys.

Table 3-101: modesChanged request keys

Key	Type	Required?	Description
appStates	array	N	Application state(s) to change. Contains dictionaries of Table 3-102 (page 166).
resources	array	N	Resource ownership(s). Contains dictionaries of Table 3-103 (page 166).

Table 3-102: appState keys

Key	Type	Required?	Description
appStateID	enum	Y	One of Table 3-86 (page 133).
entity	enum	Y	One of Table 3-81 (page 131).
speechMode	enum	N*	One of Table 3-87 (page 133). * Required if appStateID is Speech.

Table 3-103: resource keys

Key	Type	Required?	Description
resourceID	enum	Y	Identifies the resource. One of Table 3-82 (page 131).
entity	enum	Y	Identifies the current owner of the resource. Value must not be 'None'. One of Table 3-81 (page 131).
permanentEntity	enum	Y	Identifies the permanent owner of the resource. Value must not be 'None'. One of Table 3-81 (page 131).

3.3.8.6 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.

Table 3-104: forceKeyFrame request keys

Key	Type	Required?	Description
uuid	string	Y	Specifies the UUID of the display.

3.3.8.7 hidSendReport

Sender: Accessory

Description: When an HID event occurs on the accessory, such as the user turning a knob, it sends it to the 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 3-105 (page 167) lists the HID event keys.

Table 3-105: 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 device's clock).
uuid	string	Y	UUID to uniquely identify the HID device.

3.3.8.8 hidSetInputMode

Sender: Device

Description: Sets input mode on a HID.

Table 3-106: hidSetInputMode request keys

Key	Type	Required?	Description
hidInputMode	enum	Y	One of Table 3-107 (page 167).
uuid	string	Y	UUID to uniquely identify the HID device.

Table 3-107: HID input modes enum values

Name	Value	Description
Default	0	Default mode for non-character input (e.g. panning). See "3.3.5.1.4 Single Touch" (page 140).
Character	1	Optimize for entering characters. See "3.3.5.1.4 Single Touch" (page 140) and "3.3.5.2 Character Input Gesture Support" (page 142).
Scrolling	2	Optimize for non-character input (e.g. panning). See "3.3.5.1.4 Single Touch" (page 140).
ScrollingWithCharacters	3	Optimize for non-character input (e.g. panning) and entering characters. See "3.3.5.1.4 Single Touch" (page 140) and "3.3.5.2 Character Input Gesture Support" (page 142).
DialPad	4	Optimize for non-character input (e.g. panning) and entering dial-pad character events only (i.e., 0-9, #, *). See "3.3.5.1.4 Single Touch" (page 140) and "3.3.5.2 Character Input Gesture Support" (page 142), using numerical characters only.

3.3.8.9 requestSiri

Sender: Accessory

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

The requestSiri command must only be sent as a result of a direct user action on a physical or virtual control surface. The behavior should match that of "3.3.8.7 hidSendReport" (page 166), see *HID* as defined in the *Accessory Interface Specification*.

Table 3-108: requestSiri request keys

Key	Type	Required?	Description
siriAction	enum	Y	Specifies an enum value that identifies a user action. See Table 3-109 Siri actions enum values (page 168).
siriTriggerTimestamp	number	Y	DEVELOPER PREVIEW NTP timestamp indicating when the user requested Siri. See "3.3.7 Activating Siri" (page 157).
siriTriggerZone	number	N(*)	DEVELOPER PREVIEW Bitmask describing the zone in which the detection was triggered from. Table 3-62 (page 117). (*) Required if accessory supports starting Siri using voice activation.

DEVELOPER PREVIEW

Table 3-109: Siri actions enum values

Name	Value	Description
Prewarm	1	Indicate that the device should begin preparing Siri. At this point, no audio or video resources will be taken.
ButtonDown	2	Indicate to the device that the Siri button has been depressed.
ButtonUp	3	Indicate to the device that the Siri button has been released.
VoiceActivation	4	Indicate to the device that voice activation has been detected, and that the device should prepare to listen for a trigger phrase.

3.3.8.10 requestUI

Sender: Accessory or Device

Description: For the device to ask for the accessory UI to be shown, or for the accessory to ask for the CarPlay UI to be shown on the main display.

The accessory must not send requestUI upon connection; the device will send requestUI, if appropriate.

The `requestUI` command must only be sent as a result of a direct user action on a physical or virtual control surface. The accessory must not send requestUI upon connection; the device will request resources based on the modes provided in the info message. See *HID* as defined in the *Accessory Interface Specification*.

Where the sender is the accessory, a URL request key (see Table 3-110 (page 169)) is available to specify the desired CarPlay compatible app to be shown on the main display.

Where the sender is the device, a URL request key (see Table 3-111 (page 170)) is available to specify the desired accessory UI to be shown on the main display.

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

Table 3-110: Accessory to device requestUI request keys

Key	Type	Required?	Description
url	string	N	URL identifier of the desired CarPlay UI application to launch: • no url - Displays the CarPlay screen. • "app:appbundleid" - Displays a CarPlay application using the bundleID provided in the CarPlayAppList parameter group of "15.5.2 AppDiscoveryUpdate" (page 214). • "maps:" - Displays the CarPlay Maps application. • "maps:/car/destinations" - Displays the CarPlay Maps application with the Destinations view open. • "messages:" - Displays the CarPlay Messages application. • "mobilephone:" - Displays the CarPlay Phone application. • "music:" - Displays the CarPlay Music application. • "nowplaying:" - Displays the Now Playing screen. • "tel:xxx-xxx-xxxx" - Displays the CarPlay Phone application and calls the specified number. For information on supported phone number formats, see *IETF RFC 3966*.

Table 3-111: 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 that allows for re-entering the CarPlay UI. • "oem:back" - Show the last accessory screen that was visible before showing the CarPlay Ui. (Only sent when the accessory includes enhancedRequestCarUi within extendedFeatures in the Info Response, see "3.3.2.4 Info Message" (page 99).) URL identifier of the application category displayed in CarPlay: • "maps:" • "messages:" • "mobilephone:" • "nowplaying:" (Only sent when accessory includes uiContextNowOnDisplayURLs in the info message response (see "3.3.2.4 Info Message" (page 99) and "3.3.6 Shortcut Buttons" (page 155))).

3.3.8.11 setNightMode

Sender: Accessory

Description: Only if the accessory is capable of detecting whether it is night or day it must send this command to update nightMode.

Table 3-112: setNightMode request keys

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

3.3.8.12 setLimitedUI

Sender: Accessory

Description: The accessory may indicate whether or not to limit certain UI elements. The accessory may use this command only if it limits its native user interface in a similar fashion. See Table 3-43 (page 112).

Table 3-113: setLimitedUI request keys

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

3.3.8.13 updateVehicleInformation

Sender: Accessory

Description: The accessory may update the state of vehicleInformation. See Table 3-32 (page 100).

Table 3-114: vehicleInformation request keys

Key	Type	Required?	Description
vehicleInformation	group	Y	The updated vehicle information. See Table 3-48 (page 114).

3.3.8.14 iAPSendMessage

Sender: Accessory or Device

Description: Sends an iAP2 protocol message. This command must only be used for CarPlay over wireless sessions.

Table 3-115: iAPSendMessage request keys

Key	Type	Required?	Description
data	data	Y	The content of the iAP2 message.

3.3.8.15 flushAudio

Sender: Device

Description: Flush any unplayed audio data.

3.3.8.16 performHapticFeedback

Sender: Device

Description: Perform haptic feedback. Accessory perform haptics within 140 ms of receiving this command.

Table 3-116: performHapticFeedback request keys

Key	Type	Required?	Description
hapticFeedbackType	bitfield	Y	The haptic feedback type to perform, see Table 3-53 (page 115).
uuid	string	Y	The touchpad UUID, see Table 3-88 (page 137).

3.3.8.17 updateViewArea

Sender: Accessory

Description: If the device supports `viewAreas`, the accessory is required to send a `updateViewArea` command every time it changes the state of the current `viewArea` or that of `adjacentViewAreas`. The accessory must send `updateViewArea` command even when the accessory owns the screen.

Table 3-117: updateViewArea request keys

Key	Type	Required?	Description
uuid	string	Y	Specifies the UUID of the display.
viewAreaIndex	number	Y	Specifies the index of the requested view area.
adjacentViewAreas	array	Y	Specifies an array of `viewArea` indexes that can be transitioned to from the view area referenced in `viewAreaIndex`. Set to null to declare that transitioning to another view area is not possible. if the view area referenced in `viewAreaIndex` shows a transition control exactly one index must be included.
animationDurationMillis	number	Y	Specifies the duration of the animation between the active view area and the view area referenced in `viewAreaIndex` in milliseconds. This must match the animation transition period for the native system. If CarPlay is not being displayed to the user this must be set to 0.

3.3.8.18 requestViewArea

Sender: Device

Description: The CarPlay UI can be configured to present a control for the user to change the current `viewArea`. Should the user press that control device will send a `requestviewArea` command to the vehicle system. When that is received the vehicle system would respond with a `updateViewArea` to instruct the device to transition to that `viewArea`.

Table 3-118: requestViewArea request keys

Key	Type	Required?	Description
uuid	string	Y	Specifies the UUID of the display.
viewAreaIndex	number	Y	Specifies the index of the requested `viewArea`. See changes to display key.

3.3.8.19 changeUIContext

Sender: Accessory or Device

Description: Changes a UI context between the native UI and CarPlay UI.

Table 3-119: changeUIContext request keys

Key	Type	Required?	Description
url	string	Y	One string from UI context URLs included in uiContextLastOnDisplayURLs in Table 3-32 (page 100).

3.3.8.20 accessoryGiveFocus

Sender: Accessory

Description: Transfers UI focus from native UI to CarPlay.

Table 3-120: accessoryGiveFocus request keys

Key	Type	Required?	Description
originXPixels	number	Y	X coordinate of the top left corner of the focused native UI element, relative to the display.
originYPixels	number	Y	Y coordinate of the top left corner of the focused native UI element, relative to the display.
widthPixels	number	Y	Width of the focused native UI element.
heightPixels	number	Y	Height of the focused native UI element.
focusHeading	number	Y	The heading from which the native UI is providing UI focus. see Table 3-121 (page 173)

Table 3-121: focusHeading bitfield enum values

Value	Bit	Descriptions
0x01	0	Up
0x02	1	Down
0x04	2	Left
0x08	3	Right
0x10	4	Next
0x20	5	Previous

3.3.8.21 deviceOfferFocus

Sender: Device

Description: Offers the accessory to transfer focus from CarPlay to the native UI.

Table 3-122: deviceOfferFocus request keys

Key	Type	Required?	Description
originXPixels	number	Y	X coordinate of the top left corner of the focused CarPlay UI element, relative to the display.
originYPixels	number	Y	Y coordinate of the top left corner of the focused CarPlay UI element, relative to the display.
widthPixels	number	Y	Width of the focused CarPlay UI element.
heightPixels	number	Y	Height of the focused CarPlay UI element
focusHeading	number	Y	The heading from which CarPlay is providing focus. see Table 3-121 (page 173)

3.3.8.22 accessoryAcquireFocus

Sender: Accessory

Description: Transfers UI focus from CarPlay to native UI.

3.3.8.23 showUI

Sender: Accessory

Description: For the accessory to ask for a specific UI content type to be shown in an instrument cluster UI stream.

The `showUI` command must be sent with a URL request key (see Table 3-123 (page 174)) which describes which type of UI content should be shown in the display surface described by the display UUID (see Table 3-36 (page 107)).

Table 3-123: showUI request keys

Key	Type	Required?	Description
url	string	Y	DEVELOPER PREVIEW URL identifier of the desired content to be shown on the instrument cluster using the Maps URL scheme. This must inlcude one of the paths described in Table 3-124 (page 175). Optionally, to request visiblilty of certain UI elements, one or more queries described in Table 3-125 (page 175) can be appended after the path and preceded by a '?' character. Visibility of configurable UI elements are subject to app support and available space in the UI.
uuid	string	Y	Specifies the UUID of the display.

DEVELOPER PREVIEW

Table 3-124: showUI Maps URL scheme paths

Path	Description
maps:/car/instrumentcluster	Displays navigation content determined by the iOS app.
maps:/car/instrumentcluster/instructioncard	Displays a navigation instruction card.
maps:/car/instrumentcluster/map	Displays a navigation map.

Table 3-125: showUI Maps URL scheme queries

Query	Value	Description
showCompass	no	Compass not visible.
	yes	Compass visible.
	user	Compass visibility determined by user setting on device.
showETA	no	Estimated Time of Arrival (ETA) not visible.
	yes	Estimated Time of Arrival (ETA) visible.
showSpeedLimit	no	Speed limit not visible.
	yes	Speed limit visible.
	user	Speed limit visibility determined by user setting on device.

Queries must not be used with the 'car/instrumentcluster/instructioncard' path.

Examples:

If the accessory wants to show an instruction card it would include the following URL string in the showUI command:

```
maps:/car/instrumentcluster/instructioncard
```

If the accessory wants to show content with speed limits displayed it would include the following URL string in the showUI command:

```
maps:/car/instrumentcluster?showSpeedLimit=yes
```

If the accessory wants to show a map with a compass, but without speed limits displayed, it would include the following URL string in the showUI command:

```
maps:/car/instrumentcluster/map?showCompass=yes&showSpeedLimit=no
```

3.3.8.24 stopUI

Sender: Accessory

Description: For the accessory to stop UI content being drawn to an instrument cluster UI stream.

Table 3-126: stopUI request keys

Key	Type	Required?	Description
uuid	string	Y	Specifies the UUID of the display.

3.3.8.25 suggestUI

Sender: Device

Description: For the device to suggest user relevant content to be displayed on an instrument cluster.

Table 3-127: suggestUI request keys

Key	Type	Required?	Description
urls	array	Y	URLs of instrument cluster content suggestions.

3.3.8.26 uiAppearanceUpdate

Sender: Accessory

Description: For the accessory to update the current system UI appearance information for a display, see "3.2.7.1.9 Appearance Modes" (page 77).

Table 3-128: uiAppearanceUpdate request keys

Key	Type	Required?	Description
appearanceMode	enum	Y	System UI appearance mode. One of Table 3-59 (page 116).
appearanceSetting	enum	Y	System UI appearance setting. One of Table 3-60 (page 116).
uuid	string	Y	Specifies the UUID of the display.

3.3.8.27 mapAppearanceUpdate

Sender: Accessory

Description: For the accessory to update the current map appearance information for a display, see "3.2.7.1.9 Appearance Modes" (page 77).

Table 3-129: mapAppearanceUpdate request keys

Key	Type	Required?	Description
appearanceMode	enum	Y	Map appearance mode. One of Table 3-59 (page 116).
appearanceSetting	enum	Y	Map appearance setting. One of Table 3-60 (page 116).
uuid	string	Y	Specifies the UUID of the display.

3.3.8.28 changeMapZoomLevel

DEVELOPER PREVIEW

Sender: Accessory

Description: For the accessory to change the zoom level of a map being shown on an instrument cluster display.

Table 3-130: changeMapZoomLevel request keys

Key	Type	Required?	Description
zoomDirection	enum	Y	Reprsents which direction the zoom level should be changed. One of Table 3-131 (page 177).

Table 3-131: zoomDirection enum values

Value	Description
0x00	Zoom in.
0x01	Zoom out.

3.3.9 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 "3.3 CarPlay Communication Protocol" (page 92). The CarPlay Communication Plug-in is used to establish and maintain the communication and data links between the accessory and device.

The latest version of the CarPlay Communication Plug-in must be used to implement the CarPlay protocol.

The accessory maker may 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 are commands and user events. Figure 3-31 (page 179) 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:

- **AccessorySDK** provides a CoreUtils library with utilities and abstractions for OS and low-level services. This includes CoreFoundation Lite, networking utilities, and encryption/decryption implementations.
- **MFiServer** provides integration with platform specific installation of an MFi authentication chip, e.g. over I^2C 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.
- **APSAudioSession** provides information on what the platform supports for audio. This includes supported sample rates and latency values.
- **APSAudioConverter** provides integration with platform-specific audio decoders/encoders. The plug-in will route audio data to the codecs for compression and decompression. This is only required for CarPlay over wireless, where data is provided as compressed streams.
- **HIDUtils** provides a virtual HID interface to translate platform specific HID devices (touchscreen, knob-based controls, etc.) to the device. A CarPlay client application can first register the available HID devices and then post HID reports to the plug-in to be transferred to the device. Sample virtual HID devices are provided as a reference and can be customized to match the desired input device.
- **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 and input modes, commands for `requestUI` and `changeModes`, and bi-directional iAP2 communication for CarPlay over wireless.
- **APAdvertiser** provides an interface to Bonjour for configuring, querying, and advertising Bonjour services.
- Deprecated: **CarPlayControlClient** provides an interface to find and support multiple phone scenarios for both wireless and/or wired solutions. Once a device is discovered over Bonjour, `CarPlayControlClient` can be used to initiate a connection to the intended device

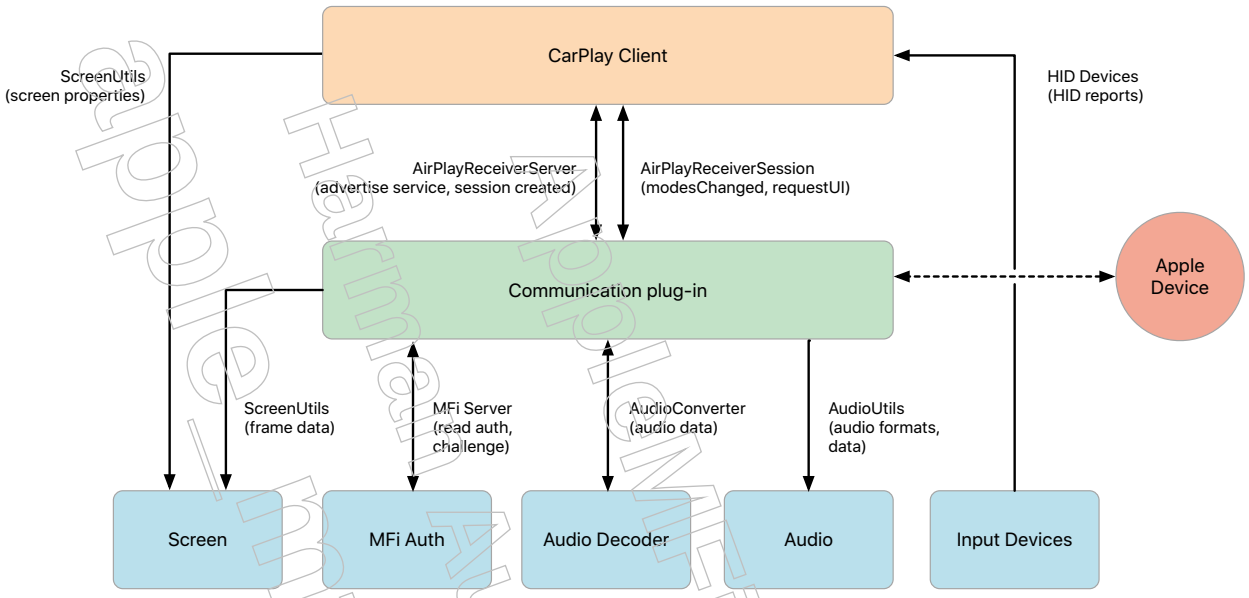

Figure 3-31: CarPlay Communication Plug-in Reference Architecture

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

3.4 Test Procedures

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

4 CarPlay Connection

CarPlay Connection is used to establish a CarPlay session with the accessory.

4.1 Requirements

All CarPlay accessories must send or receive the following iAP2 control session message(s):

- "15.6.1 CarPlayAvailability" (page 216)
- "15.6.2 CarPlayStartSession" (page 217)

4.2 Usage

CarPlay Connection is a complete replacement for Apple Bonjour zero-configuration networking. Accessories must continue to support Bonjour and the CarPlay Control command for backwards compatibility with older devices, as specified in *Accessory Interface Specification CarPlay Addendum R4*.

Refer to "3.2.4.5 Session Establishment" (page 31) for CarPlay over USB, "3.2.5.5 Session Establishment" (page 51) and "3.2.5.6 Session Reconnection" (page 53) for CarPlay over wireless.

4.3 Test Procedures

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

5 Destination Information

The destination information sharing feature provides compatible accessories with a means to receive a destination from a device for the purpose of route guidance and navigation.

This feature may only be used by CarPlay accessories that support Navigation Aided Driving. See Table 3-48 (page 114).

5.1 Requirements

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

- "15.2.1 StartDestinationInformation" (page 200)
- "15.2.2 DestinationInformation" (page 200)
- "15.2.3 DestinationInformationStatus" (page 201)
- "15.2.4 StopDestinationInformation" (page 202)

5.2 Usage

The accessory must declare a `MapsDisplayName` parameter in the Table 15-44 (page 221) component of their

"15.7.1 IdentificationInformation" (page 219) message.

Following successful identification, the accessory may send the "15.2.1 StartDestinationInformation" (page 200) message to indicate that it is ready to accept destination data. The device may then send "15.2.2 DestinationInformation" (page 200) messages until the accessory either disconnects or sends the "15.2.4 StopDestinationInformation" (page 202) message. The accessory may start and stop "15.2.2 DestinationInformation" (page 200) messages as it sees fit throughout its connection lifecycle.

After receiving a "15.2.2 DestinationInformation" (page 200) message, the accessory must attempt to forward-geocode the `Address` parameter to its native database to obtain a coordinate (latitude and longitude).

If the accessory successfully obtains a coordinate, the accessory must compare that against the `CenterCoordinate` parameter. If the coordinate is within the `CoordinateThreshold` distance from the `CenterCoordinate`, then the destination is valid.

If the destination is valid and at least one `EntryPoint` parameter is present, then the accessory must use one of the `EntryPoint` coordinate as the routable destination for the native navigation system. If multiple `EntryPoint` coordinates are present, the accessory should use the same logic as it would for native navigation to select one coordinate.

If the destination is valid, but there are no `EntryPoint` parameters, then the accessory should use that `Address` to set a routable destination for the native navigation system.

If forward-geocoding fails or the destination is not valid due to the above comparison, then the accessory should use the `CenterCoordinate` as the routable destination for the native navigation system.

The accessory must then respond with a corresponding "15.2.3 DestinationInformationStatus" (page 201) message. If the destination was not successfully used, the accessory must respond with the `FailureOccurred` parameter; otherwise, the accessory must respond with the `ParametersSuccessfullyUsed` parameter to indicate how it obtained a routable destination.

The "15.2.3 DestinationInformationStatus" (page 201) message must be sent as soon as the accessory determines if the destination can be used and must not rely on a direct user action. The message should be sent within 3 seconds of receiving the corresponding "15.2.2 DestinationInformation" (page 200) message.

5.3 Test Procedures

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

6 Route Guidance

The Route Guidance feature enables the device to provide turn-by-turn and route guidance metadata to an accessory. If an accessory presents navigation instructions using metadata provided by its built-in navigation system (on the main display, an instrument cluster, or a heads-up display), it must do the same using metadata provided by CarPlay.

6.1 Requirements

All accessories that support the Route Guidance feature via iAP2 must send or receive the following iAP2 control session messages:

- "15.3.1 StartRouteGuidanceUpdates" (page 202)
- "15.3.2 RouteGuidanceUpdate" (page 203)
- "15.3.3 RouteGuidanceManeuverInformation" (page 206)
- "15.3.4 StopRouteGuidanceUpdates" (page 210)
- "15.3.5 LaneGuidanceInformation" (page 211)

6.2 Usage

An accessory that supports Route Guidance must identify itself before it sends or receives the required messages. If the device does not support any of the Route Guidance messages, it will reject all identification attempts that include these messages.

The accessory must individually identify each display that shows route guidance data using a RouteGuidanceDisplay-Component. See Table 15-48 `RouteGuidanceDisplayComponent` parameter group (page 223).

Each RouteGuidanceDisplayComponent maps to a physical display in the vehicle. Each component includes the maximum number of characters that the display can accommodate, so that the Maps system may format text (road names, etc.) to fit optimally within the display area.

A StartRouteGuidanceUpdates (page 202) message from the accessory to the device starts the generation of RouteGuidanceUpdate (page 203) messages from the device.

Upon starting route guidance, the device sends one RouteGuidanceUpdate (page 203) message to the accessory for each RouteGuidanceDisplayComponent. These initial messages contain all of the information that the car requires to maintain route guidance state.

The device also sends multiple RouteGuidanceManeuverInformation (page 206) messages and multiple LaneGuidanceInformation (page 211) messages. The number of messages is constrained by the number of updates requested by the accessory in the `MaxGuidanceManeuverStorageCapacity` parameter and `MaxLaneGuidanceStorageCapacity` parameter of the StartRouteGuidanceUpdates (page 202) message.

Additionally, the device may send RouteGuidanceManeuverInformation (page 206) "diff" updates to dynamically update a display as the driver approaches a junction.

The accessory may discard all RouteGuidanceManeuverInformation (page 206) messages with an index that is less than that of the current set of indexes (`RouteGuidanceManeuverCurrentList`). For example, if the driver makes a wrong turn, the device may update subsequent indexed RouteGuidanceManeuverInformation (page 206) messages with the new route, or it may send new RouteGuidanceManeuverInformation (page 206) messages starting at the next available index, followed by a RouteGuidanceUpdate (page 203) message and specify a new RouteGuidanceManeuverInformation (page 206) message index as the current index.

The accessory may send a StopRouteGuidanceUpdates (page 210) messages at any time, should it wish to cease displaying or is unable to continue displaying route guidance information to the driver.

The accessory must assume a `RouteGuidanceState` of `No Route Set` until the first "15.3.2 RouteGuidanceUpdate" (page 203) message is received.

The accessory must purge all cached maneuvers whenever a "15.3.2 RouteGuidanceUpdate" (page 203) message is received with `RouteGuidanceState` set to `No Route Set`.

The information contained in a RouteGuidanceUpdate (page 203) message or RouteGuidanceManeuverInformation (page 206) message must be presented to the user within 400 ms.

The accessory must not implement logic that requires the Controller to be in the Turn-by-turn navigation app state for Route Guidance information to be displayed. To learn more, see "3.3.4 Modes" (page 131).

The accessory must not persist or use route guidance information for any other purposes than displaying route guidance to the user.

6.3 Examples

6.3.1 Normal Junction Example

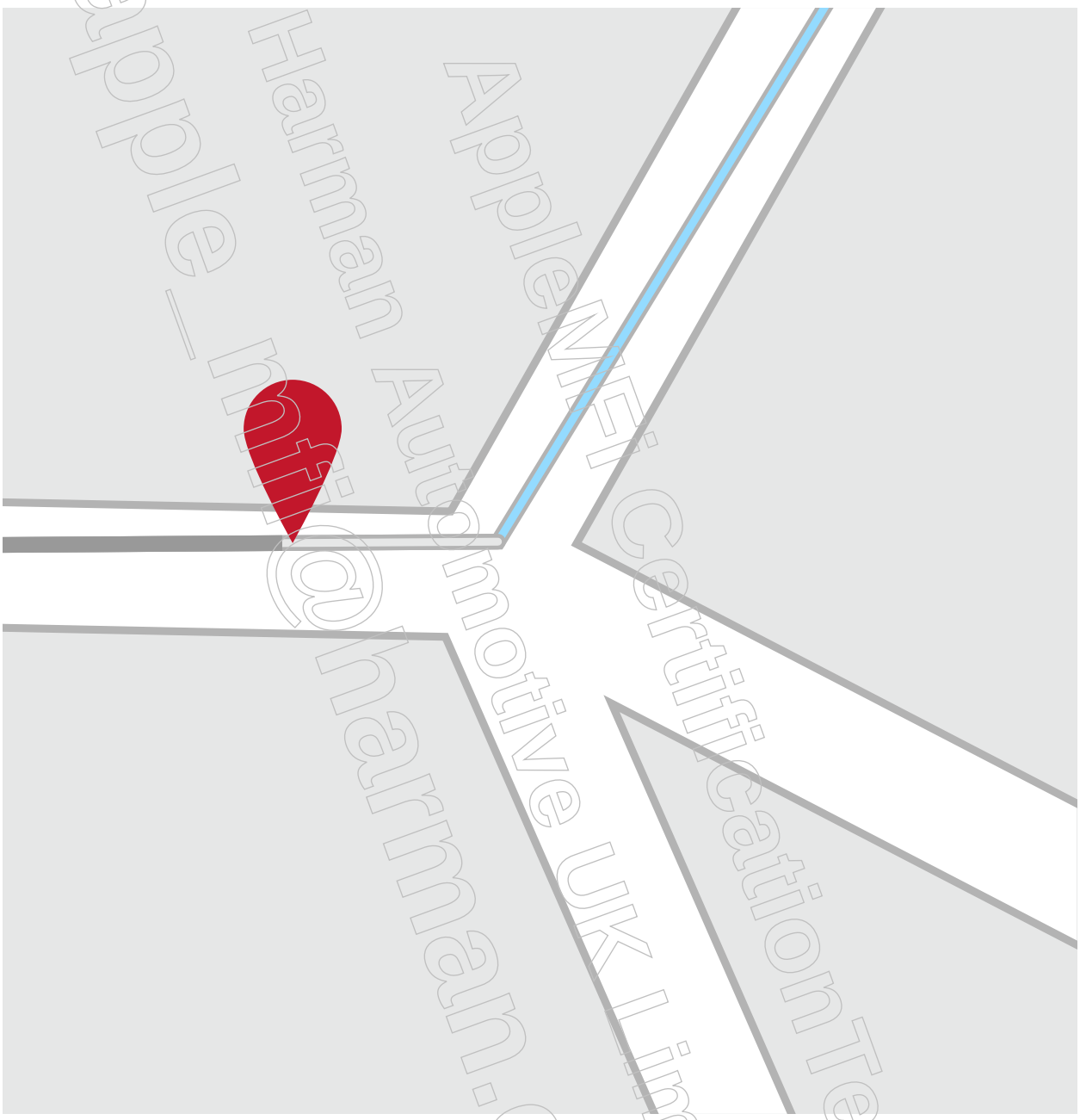

Figure 6-1: Normal Junction Element

The junction element depicted in Figure 6-1 (page 185) would have the following "15.3.3 RouteGuidanceManeuverInformation" (page 206) parameters:

- ManeuverType: 1
- DrivingSide: 0
- JunctionType: 0

- JunctionElementExitAngle: -58
- JunctionElementAngle: 28
- JunctionElementAngle: 70

6.3.2 Roundabout Junction Example

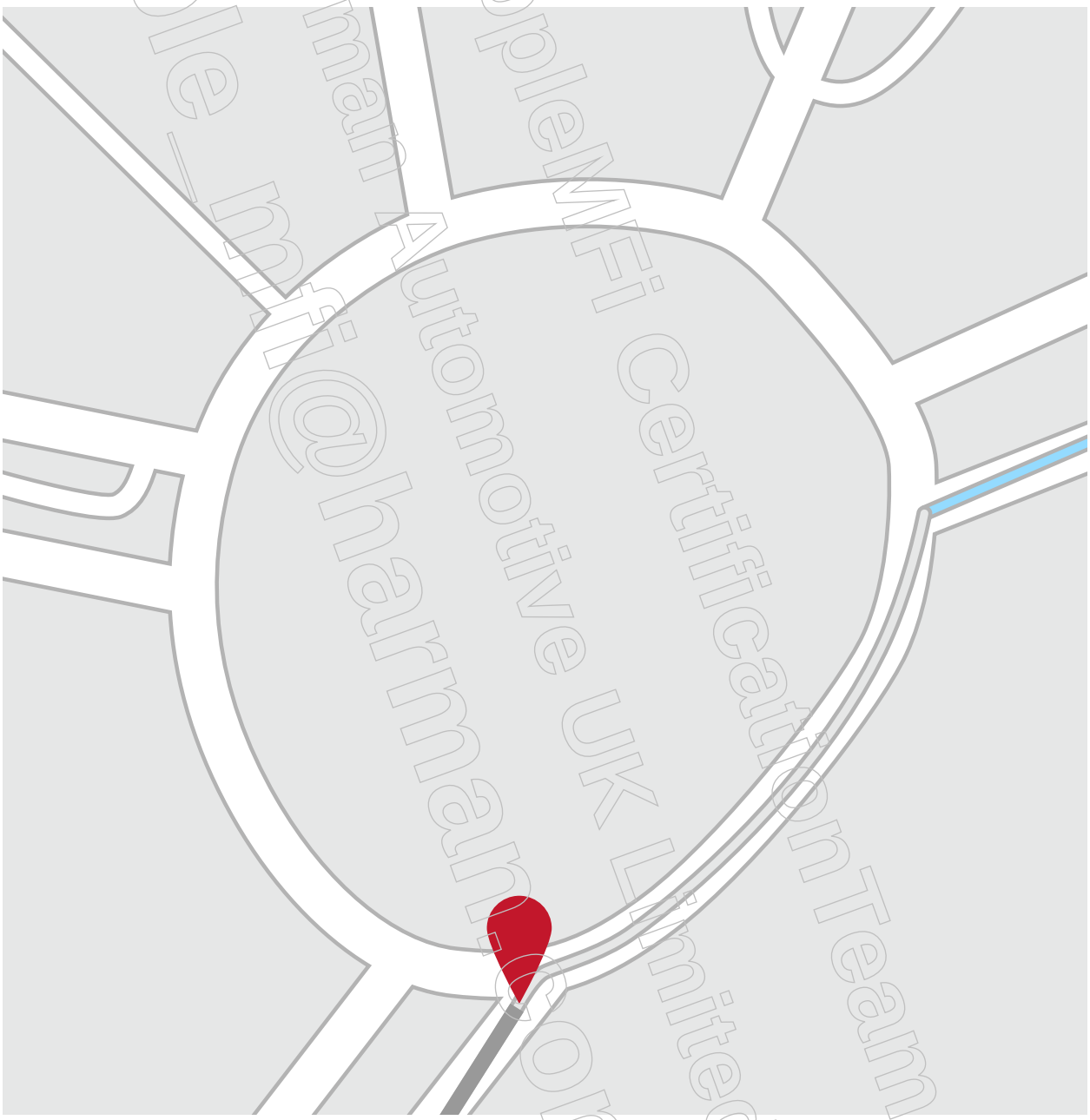

Figure 6-2: Roundabout Junction Element

The junction element depicted in Figure 6-2 (page 186) would have the following "15.3.3 RouteGuidanceManeuverInformation" (page 206) parameters:

- ManeuverType: 28
- DrivingSide: 0
- JunctionType: 1
- JunctionElementAngle: -89
- JunctionElementAngle: -54
- JunctionElementAngle: -28
- JunctionElementAngle: 19
- JunctionElementExitAngle: 64

6.3.3 Example Usage

1. The accessory identifies with one or more RouteGuidanceDisplayComponents. To learn more, see Table 15-48 `RouteGuidanceDisplayComponent` parameter group (page 223).
2. The accessory sends a StartRouteGuidanceUpdates (page 202) message start generating RouteGuidanceUpdate (page 203) messages. Route guidance may or may not be active. If route guidance is active, proceed to Step 4.
3. The device sends RouteGuidanceUpdate (page 203) message with Route Guidance information: `RouteGuidanceState` set to `NotActive`. Waits for Route Guidance to start
4. The device sends RouteGuidanceUpdate (page 203) message with Route Guidance information: `RouteGuidanceState` set to `Loading`.
5. The device sends RouteGuidanceManeuverInformation (page 206) messages with initial Route Guidance Maneuvers, up to the maximum number of maneuvers the accessory handles. `RouteGuidanceState` set to `Active`. Route guidance is active.
6. The device sends LaneGuidanceInformation (page 211) message with initial list of Lane Guidance information, up to the maximum number the accessory specified.
7. As time and driven distance continues, the device sends RouteGuidanceUpdate (page 203), RouteGuidanceManeuverInformation (page 206), and LaneGuidanceInformation (page 211) messages to update information.
    - When approaching a maneuver, the Maneuver state transitions from Continue -> Initial -> Prepare -> Execute -> Continue. The `ManeuverType` parameter specifies type of maneuver (e.g., turn right).
    - If lane guidance should be displayed, the `LaneGuidanceCurrentIndex` parameter is updated and the `LaneGuidanceShowing` parameter is set to `TRUE` to indicate that lane guidance should be displayed. When lane guidance display is no longer required, the `LaneGuidanceShowing` parameter is set to `FALSE`.
    - Repeat until route guidance ends
8. If route guidance ends, the device sends a RouteGuidanceUpdate (page 203) message with the `RouteGuidanceState` parameter set to `Arrived`.

6.4 Test Procedures

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

7 Vehicle Status

Accessories may use the Vehicle Status feature to send sensor data and other useful information from a vehicle to a device.

The following types of information can be communicated:

- Engine type
- Estimated range remaining
- Outside temperature
- Low range indicator warning status

7.1 Requirements

Accessories that implement the Vehicle Status feature must provide live information from a vehicle. Simulation of vehicle status information from other sources is not allowed.

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

- "15.4.1 StartVehicleStatusUpdates" (page 212)
- "15.4.2 VehicleStatusUpdate" (page 212)
- "15.4.3 StopVehicleStatusUpdates" (page 213)

7.2 Usage

Accessories that implement this feature must identify one Table 15-44 (page 221) and one Table 15-46 (page 222).

If the device can make use of some/all of the information, it will send a "15.4.1 StartVehicleStatusUpdates" (page 212) message to the accessory informing the accessory which information types to send updates for in a "15.4.2 VehicleStatusUpdate" (page 212) message.

The accessory must only send vehicle status updates to the device:

- Between "15.4.1 StartVehicleStatusUpdates" (page 212) and "15.4.3 StopVehicleStatusUpdates" (page 213) messages.
- Once after receiving the "15.4.1 StartVehicleStatusUpdates" (page 212) message (i.e. providing an initial value).
- When the value of a parameter has changed.
- At a maximum rate of 1 Hz.

8 Addendum: Accessory Identification

This chapter extends the corresponding chapter in the *Accessory Interface Specification*.

8.1 Requirements

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

- "15.7.1 IdentificationInformation" (page 219)
- "15.7.2 IdentificationRejected" (page 224)

8.2 Usage

For details on using these messages, see "3 CarPlay" (page 18).

8.3 Test Procedures

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

9 Addendum: App Discovery

This chapter extends the corresponding chapter in the *Accessory Interface Specification*.

The "15.5.3 RequestAppDiscoveryAppIcons" (page 215) and "15.5.4 AppDiscoveryAppIcon" (page 216) messages may only be used by accessories implementing CarPlay.

9.1 Requirements

Accessories that support the CarPlay App Discovery sub-feature via iAP2 must send or receive the following iAP2 control session messages:

- "15.5.1 StartAppDiscoveryUpdates" (page 213)
- "15.5.2 AppDiscoveryUpdate" (page 214)
- StopAppDiscoveryUpdates, see *Accessory Interface Specification*.

Accessories that support the CarPlay App Discovery sub-feature via iAP2 may send or receive the following iAP2 control session messages:

- "15.5.3 RequestAppDiscoveryAppIcons" (page 215)
- "15.5.4 AppDiscoveryAppIcon" (page 216)

The CarPlay App Discovery sub-feature enables CarPlay head units to query a device for a list of apps available in CarPlay and then open these apps in the CarPlay UI using "3.3.8.10 requestUI" (page 168). Apps available in CarPlay are grouped into categories (Audio, Calling, Messaging, Navigation, Automaker, All, etc.), allowing an accessory to retrieve a list of apps for the categories it is interested in. The retrieved app list will contain the app's app bundle ID, localized display name, category and an icon identifier for each app in the list.

If the accessory will register for icons associated with the apps, the accessory must implement an iAP2 file transfer session version 2, (see "13.1.1 File Transfer Session" (page 197)). The accessory must use the `AppDiscoveryIconData` type. All icons are transmitted to the accessory in PNG format.

The accessory must not persist or use app information for any other purposes than displaying app entry points on the screen.

9.2 Usage

A StartAppDiscoveryUpdates (page 213) message from the accessory to the device starts the generation of AppDiscoveryUpdate (page 214) messages from the device.

The accessory must include at least one category type in the `CarPlayAppCategories` parameter group in order to receive a CarPlay app list. The accessory must only include category types for apps that are shown to users. For convenience, if the accessory shows all app category types to users, the `AllCarPlayApps` category type can be used.

The `CarPlayAppListAvailable` parameter indicates whether the user has consented for CarPlay app lists to be retrieved by the accessory. The Unknown state indicates CarPlay has never used with the device. Once CarPlay is setup, a new CarPlay AppDiscoveryUpdate (page 214) message will be sent to the accessory updating the availability of the CarPlay App Discovery sub-feature, based off the preference set by the user. A CarPlay AppDiscoveryUpdate (page 214) message will also be sent out whenever the user changes the state of consent in the iOS Settings app.

Note: The `iAP2:AppIconIdentifer` parameter will not be included in the AppDiscoveryUpdate (page 214) message if an accessory does not specify an icon size in the StartAppDiscoveryUpdates (page 213) message.

An accessory may request one or more app icons using the RequestAppDiscoveryAppIcons (page 215) message. The device will respond to this request with the AppDiscoveryAppIcon (page 216) message containing the information necessary to retrieve the icon over the file transfer protocol.

In order to retrieve icons, an accessory must provide the device with its desired icon size in the StartAppDiscoveryUpdates (page 213) message (size used as both icon height and width). The device will use the icon size information to scale icons to the desired size before sending the accessory an icon identifier for each app in the app list. The icon identifier should be used to determine whether the previously retrieved icon has changed and if the icon needs to be re-retrieved. If the identifier for an app has changed, or the accessory is retrieving icons for the first time, then the accessory may initiate the retrieving of icons by sending the RequestAppDiscoveryAppIcons (page 215) message. The device will respond with an AppDiscoveryAppIcon (page 216) message for each `BundleID` in the RequestAppDiscoveryAppIcons (page 215) list. Each AppDiscoveryAppIcon (page 216) message will contain the app Bundle ID of the icon and indicate whether the icon is available. If the icon is available, the AppDiscoveryAppIcon (page 216) message will also contain the icon's file transfer ID and the icon identifier. The file transfer ID will be used by the accessory to retrieve a PNG file representing the icon over an iAP2 File Transfer session. It is recommended an accessory cache the icons for each device instead of retrieving them every time the device is connected.

An AppDiscoveryAppIcon (page 216) message with an `AppIconAvailable` parameter set to 0 (or false) indicates the app icon is unavailable from the device or the accessory is trying to retrieve an icon for a `BundleID` not contained in one of the AppDiscoveryUpdate (page 214) messages sent to the accessory.

The AppDiscoveryAppIcon (page 216) message will initiate the file transfer of the icon data over the file transfer session when the message indicates the icon is available and contains the `AppIconFileTransferID` parameter. The file transfer will not be initiated if the icon is unavailable and the AppDiscoveryAppIcon (page 216) message does not contain the `AppIconFileTransferID` parameter.

10 Addendum: Communications

This chapter extends the corresponding chapter in the *Accessory Interface Specification*.

The communications feature extends to include the following sub-features:

- **List Updates**: Allow accessories to receive the recents and favorites lists from the device.

10.1 Requirements

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

- "15.8.1 StartListUpdates" (page 224)
- "15.8.2 ListUpdate" (page 226)
- "15.8.3 StopListUpdates" (page 228)

10.2 Usage

10.2.1 List Updates

Accessories may use choose to receive "15.8.2 ListUpdate" (page 226) messages to receive the recents and favorites lists from the device as well as updates whenever these lists change. The "15.8.1 StartListUpdates" (page 224) and "15.8.3 StopListUpdates" (page 228) messages must be sent correspondingly to start or stop List Updates.

When a list update is ready to be sent from the device to the accessory, the device will first send a message with only the Available and Count (if available) parameter(s). This allows the accessory to prepare for the incoming list data by receiving the count (number of entries) in the list before the entries arrive in the messages following. If the Available parameter is false, this means the list is currently unavailable and no list entries will be sent to the accessory.

- RecentsList/RecentsListCount parameters will only be sent if the RecentsListProperties parameter was included in the "15.8.1 StartListUpdates" (page 224) message and the list is available.
- FavoritesList/FavoritesListCount parameters will only be sent if the FavoritesListProperties parameter was included in the "15.8.1 StartListUpdates" (page 224) message and the list is available.
- RecentsList -> UnixTimestamp is the timestamp of the call in represented as the number of seconds that have elapsed since 00:00:00 Coordinated Universal Time (UTC), Thursday, 1 January 1970.
- RecentsList -> Duration is the duration of the call in seconds. This parameter is only sent if RecentsList -> Occurrences is 1.
- RecentsList -> Occurrences represents the number of consecutive calls to the same person that have been combined. This parameter will always be 1 unless RecentsListCombine is True.

If the accessory displays call timestamps to the user, it should make use of the DeviceTimeUpdate message (see *Device Notifications* in the *Accessory Interface Specification*) to display the timestamp taking into account the current time zone and daylight savings offset.

The accessory may only cache list data while the device has an active iAP2 session with the accessory. The accessory must clear any cached data when the accessory is disconnected from the device. If the device sends a list update where the Available parameter is False, the accessory must clear any cached list data and display a message that the list is currently unavailable instead of any previously obtained list data.

10.3 Test Procedures

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

11 Addendum: Device Notifications

Deprecated: WirelessCarPlayUpdate and DeviceTransportIdentifierNotification are deprecated and replaced by "4 CarPlay Connection" (page 180)

This chapter extends the corresponding chapter in the *Accessory Interface Specification*.

The "15.9.1 WirelessCarPlayUpdate" (page 228) and "15.9.2 DeviceTransportIdentifierNotification" (page 229) messages may only be used by accessories implementing CarPlay over both wired and wireless transports.

11.1 Requirements

All accessories that support the Device Notifications feature via iAP2 may also send or receive the following iAP2 control session message(s):

- "15.9.1 WirelessCarPlayUpdate" (page 228)
- "15.9.2 DeviceTransportIdentifierNotification" (page 229)

11.2 Usage

For details on using these messages, see "3 CarPlay" (page 18).

11.3 Test Procedures

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

12 Addendum: Location Information

This chapter extends the corresponding chapter in the *Accessory Interface Specification*.

12.1 Requirements

All accessories that support the Location feature via iAP2 may also send or receive the following iAP2 control session message(s):

- "15.10.1 GPRMCDataStatusValuesNotification" (page 229)

12.2 Usage

The optional "15.10.1 GPRMCDataStatusValuesNotification" (page 229) message indicates which GPRMC Data status values the device can currently support.

12.3 Test Procedures

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

13 Addendum: iAP2

"9 Addendum: App Discovery" (page 191) icon downloading requires the use of File Transfer Session version 2 (see *File Transfer Session* in the *Accessory Interface Specification*.

13.1 iAP2 Session

13.1.1 File Transfer Session

13.1.1.1 Setup Datagram (Version 2)

Table 13-1: Setup Datagram File Types and File Type Setup Data message parameters

File Type	File Type Name	File Type Setup Data	Transfer Data & Note
0x0008	AppDiscoveryIconData	AppBundleID (utf8)	PNG data

14 Addendum: USB

This chapter extends the corresponding chapter in the *Accessory Interface Specification*.

14.1 USB Role Switch

14.1.1 USB Role Switch Usage

After the accessory sends the USB Custom Vendor Requests, the device will return a 4 byte data payload indicating enabled capabilities: a wValue (2 byte little endian bitfield, see Table 14-1 (page 198), followed by wIndex (2 byte little endian value). Note that wValue is encoded as a bitfield, so all Reserved values must be ignored.

Table 14-1: USB Vendor Request wValues (bitfield)

Value	Description
`0x00`	Default value
`0x01`	Device supports Apple CarPlay.
All other values	Reserved

14.2 Test Procedures

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

15 iAP2 Control Session Messages

15.1 Accessory Wi-Fi Information Sharing

Deprecated: Accessory Wi-Fi Information Sharing is deprecated and replaced by "15.6 CarPlay Connection" (page 216).

For more information, see "2 Accessory Wi-Fi Information Sharing" (page 17).

15.1.1 RequestAccessoryWiFiConfigurationInformation

Source	ID
Device	0x5702

Table 15-1: `RequestAccessoryWiFiConfigurationInformation` message parameters

Name	ID	Type	#	Notes
This message has no parameters.				

15.1.2 AccessoryWiFiConfigurationInformation

Source	ID
Accessory	0x5703

Table 15-2: `AccessoryWiFiConfigurationInformation` message parameters

Name	ID	Type	#	Notes
WiFiSSID	1	utf8	1	Wi-Fi SSID
Passphrase	2	utf8	1	Required if SecurityType is not None
SecurityType	3	enum	1	see Table 15-3 (page 200)
Channel	4	uint8	1	For 80MHz or 40MHz channel width protocols, Channel must indicate the primary Wi-Fi channel. For 20MHz channel width protocols, Channel must indicate the operating Wi-Fi channel.

Table 15-3: `AccessoryWiFiConfigurationSecurityType` enum

Value	Description
0	None
1	WEP
2	WPA2 Personal/WPA3 Personal Transition Mode

15.2 Destination Information

For more information, see "5 Destination Information" (page 181).

15.2.1 StartDestinationInformation

Source	ID
Accessory	0x10DA

Table 15-4: `StartDestinationInformation` message parameters

Name	ID	Type	#	Notes
SourceName	0	none	0/1	Must be included to receive SourceName in DestinationInformation.

15.2.2 DestinationInformation

Source	ID
Device	0x10DB

Table 15-5: `DestinationInformation` message parameters

Name	ID	Type	#	Notes
DestinationIdentifier	0	utf8	1	
DisplayName	1	utf8	0/1	
CenterCoordinate	2	group	1	See Table 15-6 (page 201)
Address	3	utf8	1	lines separated by newline characters
EntryPoint	4	group	0+	See Table 15-6 (page 201)
CoordinateThreshold	5	uint32	1	in meters
Locale	6	utf8	1	
SourceName	7	utf8	0/1	Name of CarPlay application providing the destination, for display purposes only. Do not use for basing any behavior change decisions.

Table 15-6: `Coordinate` parameter group

Name	ID	Type	#	Notes
Latitude	0	int32	1	Fixed-point representation of decimal degrees in Q10.22 format. (The most significant 10 bits contain the integer portion, and the remaining least significant 22 bits contain an integer representing the decimal portion.)
Longitude	1	int32	1	Fixed-point representation of decimal degrees in Q10.22 format. (The most significant 10 bits contain the integer portion, and the remaining least significant 22 bits contain an integer representing the decimal portion.)

15.2.3 DestinationInformationStatus

Source	ID
Accessory	0x10DD

Table 15-7: `DestinationInformationStatus` message parameters

Name	ID	Type	#	Notes
DestinationIdentifier	0	utf8	1	See "15.2.2 DestinationInformation" (page 200)
FailureOccurred	1	none	0/1	
ParametersSuccessfullyUsed	2	group	0/1	See Table 15-8 (page 202)

Table 15-8: `ParametersSuccessfullyUsed` parameter group

Name	ID	Type	#	Notes
CenterCoordinate	2	none	0/1	
Address	3	none	0/1	
EntryPoint	4	none	0/1	

15.2.4 StopDestinationInformation

Source	ID
Accessory	0x10DC

Table 15-9: `StopDestinationInformation` message parameters

Name	ID	Type	#	Notes
This message has no parameters.				

15.3 Route Guidance

For more information, see "6 Route Guidance" (page 183).

15.3.1 StartRouteGuidanceUpdates

Source	ID
Accessory	0x5200

Table 15-10: `StartRouteGuidanceUpdates` message parameters

Name	ID	Type	#	Notes
RouteGuidanceDisplayComponentID	0	uint16	0+	Specifies a component in Table 15-48 `RouteGuidanceDisplayComponent` parameter group (page 223). If none are specified, assume all such components are being started.
SourceName	1	none	0/1	Must be included to receive SourceName in RouteGuidanceUpdate (page 203).
SourceSupportsRouteGuidance	2	none	0/1	Must be included to receive SourceSupportsRouteGuidance in RouteGuidanceUpdate (page 203).
SupportsExitInfo	3	none	0/1	Must be included to receive ExitInfo in RouteGuidanceManeuverInformation (page 206).

15.3.2 RouteGuidanceUpdate

Source	ID
Device	0x5201

Table 15-11: `RouteGuidanceUpdate` message parameters

Name	ID	Type	#	Notes
RouteGuidanceDisplayComponentID	0	uint16	0+	Must refer to an identified *RouteGuidanceDisplayComponent*. If none is specified, it means the update applies to all components. If more than one are specified, update is applied to all specified components.
RouteGuidanceState	1	enum	0/1	Specifies the current RouteGuidance state, see Table 15-12 (page 206).
ManeuverState	2	enum	0/1	See Table 15-13 (page 206).
CurrentRoadName	3	utf8	0/1	The name of the current road.
DestinationName	4	utf8	0/1	The name of the destination.

EstimatedTimeOfArrival	5	uint64	0/1	Specifies the number of seconds from reference date (Jan 1, 1970 GMT). Similar to `DeviceTimeUpdate`. Accessory should use DeviceTime-Update and use the TimeZoneOffset-Minutes and DaylightSavingsOffset-Minutes information to apply the adjustment necessary for local time.
TimeRemainingToDestination	6	uint64	0/1	The number of seconds to the destination.
DistanceRemaining	7	uint32	0/1	Meters to end of trip. This value must not be displayed to the user in text. It is only to be used for generating graphics, e.g. countdown bars.
DistanceRemainingDisplayString	8	utf8	0/1	Provides a display string representation of DistanceRemaining in DistanceRemainingDisplayUnits units. Does not contain the units string. Must be displayed without unit conversion to the user.
DistanceRemainingDisplayUnits	9	enum	0/1	Units that must accompany DistanceRemainingDisplayString when displayed to the user, see Table 15-14 (page 206).
DistanceRemainingToNextManeuver	10	uint32	0/1	Meters to next maneuver. This value must not be displayed to the user as text. It is only to be used for generating graphics, e.g. countdown bars.
DistanceRemainingToNextManeuverDisplayString	11	utf8	0/1	Display string representation of DistanceToNextManeuver in DistanceToNextManeuverDisplayUnits units. Does not contain the units string. Must be displayed without unit conversion to the user.
DistanceRemainingToNextManeuverDisplayUnits	12	enum	0/1	Units that must accompany DistanceToNextManeuverDisplayString when displayed to the user, see Table 15-14 (page 206).

RouteGuidanceManeuverCurrentList	13	uint16[]	0/1	Indexes of next upcoming maneuvers in the Route Guidance maneuver list. Multiple maneuvers that occur very close together can be grouped into an ordered list of indexes with this parameter. Normally there would only be one entry for the current upcoming maneuver. If more than one item is present in RouteGuidanceManeuverCurrentList the maneuvers should be executed in quick succession and can be displayed next to one another on the display.
RouteGuidanceManeuverTotalCount	14	uint16	0/1	Total number of Maneuvers in the trip.
RouteGuidanceVisibleInApp	15	bool	0/1	Indicates whether route guidance is being show in the currently navigating app. When false, although the route guidance is not being shown in the app, RouteGuidanceUpdate and RouteGuidanceManeuverInfoUpdate messages will still be sent to the accessory.
LaneGuidanceCurrentIndex	16	uint16	0/1	Index of the LaneGuidanceInformation instruction to display.
LaneGuidanceTotalCount	17	uint16	0/1	Total number of lane guidance instructions for the trip.
LaneGuidanceBeingShownInApp	18	bool	0/1	Indicates whether lane guidance is being show in the currently navigating app.
SourceName	19	utf8	0/1	Name of the application providing turn-by-turn information in CarPlay - for display purposes only. Do not use for behavior change decisions.
SourceSupportsRouteGuidance	20	bool	0/1	Specifies whether the application providing turn-by-turn information in CarPlay can populate RouteGuidanceManeuverInformation, LaneGuidanceInformation or parameters 0–18 in RouteGuidanceUpdate.

Note: This is an update so information can be sent all together or individually and the accessory should keep a cache of the information and update as the information arrives.

Table 15-12: `RouteGuidanceState` enum

Value	Description
0	No Route Set. A route is not set.
1	Route Set. A route is set.
2	Arrived. Arrived at the destination.
3	Loading. Retrieving routing information.
4	Locating. Location/position not available.
5	Rerouting. Recalculating the route.
6	Proceed to Route. Vehicle needs to join the route.

Table 15-13: `ManeuverState` enum

Value	Description
0	Continue. Continue along this route until next maneuver.
1	Initial. Notification that maneuver is in the near future.
2	Prepare. Notification that maneuver is in the immediate future.
3	Execute. Vehicle is in maneuver.

Table 15-14: `DistanceRemainingDisplayUnits` enum

Value	Description
0	km
1	miles
2	m
3	yards
4	ft

15.3.3 RouteGuidanceManeuverInformation

Source	ID
Device	0x5202

Table 15-15: `RouteGuidanceManeuverInformation` message parameters

Name	ID	Type	#	Notes
RouteGuidanceDisplayComponentID	0	uint16	1	Refers to a component in Table 15-48 `RouteGuidanceDisplayComponent` parameter group (page 223).
Index	1	uint16	1	Specifies which item in the list is being updated.
ManeuverDescription	2	utf8	0/1	
ManeuverType	3	enum	0/1	SeeTable 15-16 (page 208).
AfterManeuverRoadName	4	utf8	0/1	The name of the road the vehicle will be on after the maneuver is completed successfully.
DistanceBetweenManeuver	5	uint32	0/1	Meters between previous maneuver to this maneuver. This value must not be displayed to the user in text. It is only to be used for generating graphics, e.g. countdown bars.
DistanceBetweenManeuverDisplayString	6	utf8	0/1	Display string representation of DistanceBetweenManeuver in DistanceBetweenManeuverDisplayUnits units. Does not contain the units string. Must be displayed without unit conversion to the user.
DistanceBetweenManeuverDisplayUnits	7	enum	0/1	Units that must accompany DistanceBetweenManeuverDisplayString when displayed to the user. See Table 15-14 (page 206).
DrivingSide	8	enum	0/1	Whether cars should drive on right or left side of the road. For roundabout, specifies the direction of traffic around the roundabout. See Table 15-17 (page 210).
JunctionType	9	enum	0/1	See Table 15-18 `JunctionType` enum (page 210).
JunctionElementAngle	10	int16	0+	Angle of a junction element (+/- 180). Accessory must show all junction elements together. An entry junction element is implied at angle 180.
JunctionElementExitAngle	11	int16	0/1	Angle of an exit junction element (+/- 180). Junction element describing the road used to exit the junction, or exit of the maneuever. Accessory must show all junction elements together. An entry junction element is implied at angle 180.
LinkedLaneGuidanceInformation	12	uint16	0/1	Specifies the index of the `LaneGuidanceInformation` associated with this maneuver.

ExitInfo	13	utf8	0/1	Information about highway exit (e.g. exit number) if available. Only sent if SupportsExitInfo is included in "15.3.1 StartRouteGuidanceUpdates" (page 202).

Note: If a RouteGuidanceManeuverInformation message is received with an index matching a maneuver already stored on the accessory, the stored maneuver must be removed and replaced by the information in that RouteGuidanceManeuverInformation message.

Note: Each message contains one RouteGuidance maneuver item in the RouteGuidance maneuver list. Index specifies which item in the list is being updated. If the index specifies an entry outside of the MaxGuidanceManeuverStorageCapacity, the oldest entry should be deleted and the new entry added. The set of junction elements describe the junction where the maneuver needs to take place. The entry junction element is implied at angle 180 and is not included as a JunctionElementAngle. Multiple junction elements for a maneuver must be shown together. Junction elements may not be provided for a maneuver and should be used to supplement the ManeuverType when present.

Table 15-16: `ManeuverType` enum

Value	Description
0	No Turn (default value)
1	Left Turn. Angle is between -45° and -135°
2	Right Turn. Angle is between 45° and 135°
3	Straight Ahead. Continue straight through the intersection (implies a road name will change)
4	Make a U-turn
5	Continue to follow the road the vehicle is currently on
6	Enter Roundabout
7	Exit Roundabout
8	Off Ramp. Take the ramp to leave the highway
9	On Ramp. Take the ramp to merge onto the highway
10	Arrive End Of Navigation. Navigation has completed, but the rest of the journey will need to use another transport method
11	Proceed to the beginning of the route
12	Arrive At Destination. Destination has been reached, navigation will end
13	Keep Left. Usually for bifurcations or other smooth maneuvers (compare to Slight Left Turn)
14	Keep Right. Usually for bifurcations or other smooth maneuvers (compare to Slight Right Turn)
15	Enter Ferry
16	Exit Ferry

17	Change to a different ferry
18	Make a U-turn and proceed to the route
19	Use the roundabout to make a U-turn
20	At the end of the road, turn left
21	At the end of the road, turn right
22	Highway Off Ramp Left. Exit the highway on the left
23	Highway Off Ramp Right. Exit the highway on the right
24	Arrive At Destination Left. Destination has been reached; it is on the left and navigation will end
25	Arrive At Destination Right. Destination has been reached; it is on the right and navigation will end
26	Make a U-turn when possible
27	Arrive End Of Directions. Navigation has completed, but the rest of the journey will need to use another transport method
28	Roundabout Exit 1. Exit the roundabout at the 1st street
29	Roundabout Exit 2. Exit the roundabout at the 2nd street
30	Roundabout Exit 3. Exit the roundabout at the 3rd street
31	Roundabout Exit 4. Exit the roundabout at the 4th street
32	Roundabout Exit 5. Exit the roundabout at the 5th street
33	Roundabout Exit 6. Exit the roundabout at the 6th street
34	Roundabout Exit 7. Exit the roundabout at the 7th street
35	Roundabout Exit 8. Exit the roundabout at the 8th street
36	Roundabout Exit 9. Exit the roundabout at the 9th street
37	Roundabout Exit 10. Exit the roundabout at the 10th street
38	Roundabout Exit 11. Exit the roundabout at the 11th street
39	Roundabout Exit 12. Exit the roundabout at the 12th street
40	Roundabout Exit 13. Exit the roundabout at the 13th street
41	Roundabout Exit 14. Exit the roundabout at the 14th street
42	Roundabout Exit 15. Exit the roundabout at the 15th street
43	Roundabout Exit 16. Exit the roundabout at the 16th street
44	Roundabout Exit 17. Exit the roundabout at the 17th street
45	Roundabout Exit 18. Exit the roundabout at the 18th street
46	Roundabout Exit 19. Exit the roundabout at the 19th street

47	Sharp Left Turn. Angle is between -135° and -180°
48	Sharp Right Turn. Angle is between 135° and 180°
49	Slight Left Turn. Turn onto a different road (compare to Keep Left)
50	Slight Right Turn. Turn onto a different road (compare to Keep Right)
51	Change Highway. Highway to highway change with implied or unknown side of the road
52	Change Highway Left. Highway to highway change from the left side of the road
53	Change Highway Right. Highway to highway change from the right side of the road

Table 15-17: `DrivingSide` enum

Value	Description
0	Right (or Anti-Clockwise for roundabouts).
1	Left (or Clockwise for roundabouts).

Table 15-18: `JunctionType` enum

Value	Description
0	Single intersection with junction elements representing roads coming to a common point.
1	Roundabout with junction elements representing roads exiting the roundabout.

15.3.4 StopRouteGuidanceUpdates

Source	ID
Accessory	0x5203

Table 15-19: `StopRouteGuidanceUpdates` message parameters

Name	ID	Type	#	Notes
RouteGuidanceDisplayComponentID	0	uint16	0+	Must refer to an identified Table 15-48 (page 223). If none are specified, assume all such components are to be stopped.

15.3.5 LaneGuidanceInformation

Source	ID
Device	0x5204

Table 15-20: `LaneGuidanceInformation` message parameters

Name	ID	Type	#	Notes
RouteGuidanceDisplayComponentID	0	uint16	1	Refers to an identified `RouteGuidanceDisplayComponent`.
LaneGuidanceIndex	1	uint16	1	Specifies which item in the list is being updated.
LaneInformation	2	group	1+	Specifies information about each lane. See Table 15-21 (page 211).
LaneGuidanceDescription	3	utf8	0/1	

Table 15-21: `LaneInformation` message parameters

Name	ID	Type	#	Notes
Index	0	uint16	1	Specifies the lane position from left (0) to right.
LaneStatus	1	enum	1	Describes if the lane should be used. See Table 15-22 (page 211).
LaneAngle	2	int16	0+	Specifies an angle between -180 and +180 degrees.
LaneAngleHighlight	3	int16	0/1	Specifies the lane angle to be highlighted.

Table 15-22: `LaneStatus` enum

Value	Description
0	Not Good. The vehicle should not take this lane.
1	Good. The vehicle can take this lane, but may need to move lanes again before upcoming maneuvers.
2	Preferred. The vehicle should take this lane to be in the best position for upcoming maneuvers.

15.4 Vehicle Status

15.4.1 StartVehicleStatusUpdates

Source	ID
Device	0xA100

Table 15-23: `StartVehicleStatusUpdates` message parameters

Name	ID	Type	#	Notes
Range	3	none	0/1	
OutsideTemperature	4	none	0/1	
RangeWarning	6	none	0/1	
RangeGasoline	9	none	0/1	
RangeDiesel	10	none	0/1	
RangeElectric	11	none	0/1	
RangeCNG	12	none	0/1	
RangeWarningGasoline	13	none	0/1	
RangeWarningDiesel	14	none	0/1	
RangeWarningElectric	15	none	0/1	
RangeWarningCNG	16	none	0/1	

15.4.2 VehicleStatusUpdate

Source	ID
Accessory	0xA101

Table 15-24: `VehicleStatusUpdate` message parameters

Name	ID	Type	#	Notes
Range	3	uint16	0/1	Remaining vehicle range in kilometers
OutsideTemperature	4	int16	0/1	measured outside temperature in °C
RangeWarning	6	bool	0/1	if True, the vehicle's low range warning indicator is set
RangeGasoline	9	uint16	0/1	Remaining vehicle range in kilometers using gasoline
RangeDiesel	10	uint16	0/1	Remaining vehicle range in kilometers using diesel
RangeElectric	11	uint16	0/1	Remaining vehicle range in kilometers using electricity
RangeCNG	12	uint16	0/1	Remaining vehicle range in kilometers using CNG
RangeWarningGasoline	13	bool	0/1	if True, the vehicle's low range warning indicator for gasoline is set
RangeWarningDiesel	14	bool	0/1	if True, the vehicle's low range warning indicator for diesel is set
RangeWarningElectric	15	bool	0/1	if True, the vehicle's low range warning indicator for electricity is set
RangeWarningCNG	16	bool	0/1	if True, the vehicle's low range warning indicator for CNG is set

15.4.3 StopVehicleStatusUpdates

Source	ID
Device	0xA102

Table 15-25: `StopVehicleStatusUpdates` message parameters

Name	ID	Type	#	Notes
This message has no parameters.				

15.5 App Discovery

For more information, see "9 Addendum: App Discovery" (page 191).

15.5.1 StartAppDiscoveryUpdates

Source	ID
Accessory	0xAD00

Table 15-26: `StartAppDiscoveryUpdates` message parameters

Name	ID	Type	#	Notes
CarPlayAppCategories	0	group	1	Specifies one or more categories of CarPlay UI apps. The accessory must specify at least one category to receive a CarPlay app list. For more information, see Table 15-27 (page 214).
CarPlayAppListMax	1	uint16	0/1	Specifies the maximum number of CarPlay apps returned in the app list. By default, 0 = no limit.
CarPlayAppIconSize	2	uint16	0/1	Specifies the size of the CarPlay UI app icon in pixels. The value specified defines the height and width of the app icon. For example, a value of 90 returns an app icon that measures 90x90 pixels. Maximum value is 180.

Table 15-27: `CarPlayAppDiscoveryCategories` parameter group

Name	ID	Type	#	Notes
AllCarPlayApps	0	none	0/1	The All CarPlay Apps category returns CarPlay apps across all functional areas.
Messaging	1	none	0/1	
Calling	2	none	0/1	
Navigation	3	none	0/1	
Audio	4	none	0/1	
Automaker	5	none	0/1	
QuickOrdering	7	none	0/1	
EVCharging	8	none	0/1	
Parking	9	none	0/1	

15.5.2 AppDiscoveryUpdate

Source	ID
Device	0xAD01

Table 15-28: `AppDiscoveryUpdate` message parameters

Name	ID	Type	#	Notes
CarPlayAppListAvailable	0	enum	1	Indicates availability CarPlay app lists. For more information, see Table 15-29 (page 215).
CarPlayAppList	1	group	0+	A group that specifies an app list of CarPlay UI apps and the accompanying app icons. For more information, see Table 15-30 (page 215).
CarPlayAppListCount	2	uint16	0/1	Specifies the number of entries in the CarPlay app list.

Table 15-29: `CarPlayAppListAvailable` enum

Value	Description
0	Unknown. Returned if CarPlay has not been set up on the device. Indicates that this is the first connection to an accessory and that the user will be prompted to set up CarPlay.
1	Declined. Returned if user did not consent to app lists being shared with the accessory.
2	Accepted. Returned if user consented to app lists being shared with the accessory.

Table 15-30: `AppList` parameter group

Name	ID	Type	#	Notes
AppIconIdentifier	4	blob	0/1	Specifies the identifier for the app's icon. The `AppIconIdentifer` is not returned in the `StartAppDiscoveryUpdates` message if the accessory does not specify the size of the icon in the `StartAppDiscoveryUpdates` message. To learn more, see Table 15-32 (page 216).

15.5.3 RequestAppDiscoveryAppIcons

Source	ID
Accessory	0xAD03

Table 15-31: `RequestAppDiscoveryAppIcons` message parameters

Name	ID	Type	#	Notes
CarPlayAppBundleID	0	utf8	0+	Specifies the bundle ID of the CarPlay app.

15.5.4 AppDiscoveryAppIcon

Source	ID
Device	0xAD04

App icon files are not transferred if the icon is unavailable or the `AppIconFileTransferID` parameter is unspecified. If no app icon size is specified in the `StartAppDiscoveryUpdates` message, the `AppIconAvailable` parameter is set to false.

Table 15-32: `AppDiscoveryAppIcon` message parameters

Name	ID	Type	#	Notes
AppBundleID	0	utf8	1	Specifies the app's bundle ID, a Uniform Type Identfier in reverse-DNS format, e.g. `com.ajax.hello`.
AppIconAvailable	1	bool	1	Indicates whether an app icon is available to download: (1: True, 0: False). If false, the app icon is either unavailable from the device or the accessory is attempting to download an app icon that was not specified in the app list.
AppIconFileTransferID	2	uint8	0/1	Specifies the app icon's file transfer ID. The accessory may use the file transfer ID to download app icon PNG files over an iAP2 File Transfer session.
AppIconIdentifier	3	blob	1	Specifies the app icon's ID. Enables the accessory to determine whether a previously downloaded app icon has been updated and needs to be re-downloaded.

15.6 CarPlay Connection

For more information, see "4 CarPlay Connection" (page 180).

15.6.1 CarPlayAvailability

Source	ID
Device	0x4300

CarPlayAvailability indicates if CarPlay is enabled on the device. CarPlayAvailability is always sent over the USB transport. Over Bluetooth, during initial pairing for wireless CarPlay, CarPlayAvailability is sent only after the user has confirmed and enabled wireless CarPlay on the device. CarPlayAvailability is always sent over Bluetooth during reconnections.

Table 15-33: `CarPlayAvailability` message parameters

Name	ID	Type	#	Notes
WiredAttributes	0	group	0/1	Indicates availability of CarPlay over USB. May be absent, if the message is sent over Bluetooth and the user has disabled wireless CarPlay on the device, see Table 15-34 (page 217).
WirelessAttributes	1	group	0/1	Indicates availability of CarPlay over wireless. May be absent, if the message is sent over USB and the device has not been paired yet, see Table 15-35 (page 217)

Table 15-34: `CarPlayAvailabilityWiredAttributes` parameter group

Name	ID	Type	#	Notes
Available	0	bool	1	
USBTransportIdentifier	1	utf8	0/1	If included, represents a USB transport identifier. Absent if value of Available is false.

Table 15-35: `CarPlayAvailabilityWirelessAttributes` parameter group

Name	ID	Type	#	Notes
Available	0	bool	1	
BluetoothTransportIdentifier	1	utf8	0/1	If included, represents an Bluetooth transport identifier. Absent if value of Available is false.

15.6.2 CarPlayStartSession

Source	ID
Accessory	0x4301

Table 15-36: `CarPlayStartSession` message parameters

Name	ID	Type	#	Notes
WiredAttributes	0	group	0/1*	* Must included when message is sent over USB transport, see Table 15-37 (page 218)
WirelessAttributes	1	group	0/1*	* Must be included when message is sent over Bluetooth transport, see Table 15-38 (page 218)
Port	2	uint32	1	
DeviceIdentifier	3	utf8	1	Globally unique ID for the accessory; e.g. the primary MAC address, such as "00:11:22:33:44:55". Must match the value of deviceID in Info message response, see Table 3-32 (page 100).
PublicKey	4	utf8	1	Pairing identity string. See "3.3.2.1 Pairing" (page 97)
SourceVersion	5	utf8	1	Apple CarPlay communication plug-in source version assigned by Apple, in the form "x.y.z".

Table 15-37: `CarPlayStartSessionWiredAttributes` parameter group

Name	ID	Type	#	Notes
IPAddress	0	utf8	1	Accessory's link-local IPv6 address for wired interface. IPv6 address must not include a zone index.

Table 15-38: `CarPlayStartSessionWirelessAttributes` parameter group

Name	ID	Type	#	Notes
WiFiSSID	0	utf8	1	
Passphrase	1	utf8	1	
Channel	2	uint8	1	For 80MHz or 40MHz channel width protocols, Channel must indicate the primary Wi-Fi channel. For 20MHz channel width protocols, Channel must indicate the operating Wi-Fi channel.
IPAddress	3	utf8	1	Accessory's link-local IPv6 address for Wi-Fi interface. IPv6 address must not include a zone index.
SecurityType	4	uint8	1	DEVELOPER PREVIEW See Table 15-39 (page 219)

DEVELOPER PREVIEW

Table 15-39: CarPlayStartSession Wi-Fi SecurityType enum

Value	Description
0	Reserved
1	Reserved
2	Reserved
3	WPA3 Personal Transition Mode
4	WPA3 Personal Only

15.7 Addendum: Accessory Identification

For more information, see *Accessory Identification* in the *Accessory Interface Specification*.

15.7.1 IdentificationInformation

Source	ID
Accessory	0x1D01

Table 15-40: `IdentificationInformation` message parameters

Name	ID	Type	#	Notes
Name	0	utf8	1	Shall reflect the marketing name of the accessory.
ModelIdentifier	1	utf8	1	Shall reflect the vehicle model.
Manufacturer	2	utf8	1	Shall reflect the manufacturer of the accessory or vehicle (Tier 1 supplier or OEM).
SerialNumber	3	utf8	1	Shall reflect accessory's serial number. This must match the serial number displayed in the accessory's UI (if applicable).
FirmwareVersion	4	utf8	1	Shall reflect the current revision of the accessory's firmware. This must match the firmwareRevision key in the info message response if provided (see "3.3.2.4 Info Message" (page 99)), as well as any software/firmware version displayed in the accessory's UI (if applicable).
HardwareVersion	5	utf8	1	Shall reflect the current revision of the accessory's hardware. This must match the hardwareRevision key in the info message response if provided (see "3.3.2.4 Info Message" (page 99)), as well as any hardware version displayed in the accessory's UI (if applicable).
VehicleInformationComponent	20	group	0/1	See Table 15-44 (page 221).
VehicleStatusComponent	21	group	0/1	See Table 15-46 (page 222).
WirelessCarPlayTransportComponent	24	group	0+	See Table 15-47 (page 222).
RouteGuidanceDisplayComponent	30	group	0+	See Table 15-48 (page 223).

Note: Requirements provided for parameters 0–5 in the *Notes* column in Table 15-40 (page 220) override requirements for those IdentificationInformation message parameters in *Accessory Interface Specification*.

Table 15-41: `ExternalAccessoryProtocol` parameter group

Name	ID	Type	#	Notes
ExternalAccessoryProtocolCarPlay	4	none	0/1	Must only be set by a CarPlay accessory where this protocol is intended to match with an app that supports CarPlay.

Table 15-42: `MatchAction` enum

Value	Description
4	No Action and No Communication Protocol. The protocol is not intended for communication and does not require StartExternalAccessoryProtocolSession or StopExternalAccessoryProtocolSession (see *External Accessory Protocol, Accessory Interface Specification*). The device will not prompt the user to find a matching app, and there will not be a Find App For This Accessory button in Settings > General > About > 'Accessory Name' (see *App Match, Accessory Interface Specification*).

Table 15-43: `USBHostTransportComponent` parameter group

Name	ID	Type	#	Notes
USBHostTransportCarPlayInterfaceNumber	3	uint8	1	The accessory's NCM Control Interface number.
TransportSupportsCarPlay	4	none	1	

Table 15-44: `VehicleInformationComponent` parameter group

Name	ID	Type	#	Notes
Identifier	0	uint16	1	All Identifiers must be unique.
Name	1	utf8	1	
EngineType	2	enum	0+	Presence of this parameter indicates that the vehicle can externally replenish the given engine energy source. For example, a hybrid electric vehicle that cannot be charged externally must not declare an Electric EngineType. See Table 15-45 (page 221).
DisplayName	6	utf8	1	
MapsDisplayName	8	utf8	0/1	

Table 15-45: `EngineTypes` enum

Value	Description
0	Gasoline
1	Diesel
2	Electric
3	CNG

Table 15-46: `VehicleStatusComponent` parameter group

Name	ID	Type	#	Notes
Identifier	0	uint16	1	All Identifiers must be unique.
Name	1	utf8	1	
Range	3	none	0/1	
OutsideTemperature	4	none	0/1	
RangeWarning	6	none	0/1	Include for vehicles with a single EngineType, or vehicles with multiple EngineTypes that report a unified range warning for all EngineTypes.
RangeGasoline	9	none	0/1	
RangeDiesel	10	none	0/1	
RangeElectric	11	none	0/1	
RangeCNG	12	none	0/1	
RangeWarningGasoline	13	none	0/1	Include for vehicles that support Gasoline EngineType that show a specific range warning for Gasoline. Do not include if vehicle reports unified range warning for all EngineTypes.
RangeWarningDiesel	14	none	0/1	Include for vehicles that support Diesel EngineType that show a specific range warning for Diesel. Do not include if vehicle reports unified range warning for all EngineTypes.
RangeWarningElectric	15	none	0/1	Include for vehicles that support Electric EngineType that show a specific range warning for Electric. Do not include if vehicle reports unified range warning for all EngineTypes.
RangeWarningCNG	16	none	0/1	Include for vehicles that support CNG EngineType that show a specific range warning for CNG. Do not include if vehicle reports unified range warning for all EngineTypes.

Table 15-47: `WirelessCarPlayTransportComponent` parameter group

Name	ID	Type	#	Notes
TransportComponentIdentifier	0	uint16	1	All TransportComponent identifiers must be unique
TransportComponentName	1	utf8	1	
TransportSupportsiAP2Connection	2	none	1	
TransportSupportsCarPlay	4	none	1	

Table 15-48: `RouteGuidanceDisplayComponent` parameter group

Name	ID	Type	#	Notes
Identifier	0	uint16	1	All Identifiers must be unique.
Name	1	utf8	1	
MaxCurrentRoadNameLength	2	uint16	0/1	How many of the widest character in the font used to display CurrentRoadName can be shown to the user. Must be included if CurrentRoad-Name information is displayed to the user. Not including this parameter or setting a value of 0 indicates that CurrentRoadName is not displayed to the user.
MaxDestinationNameLength	3	uint16	0/1	How many of the widest character in the font used to display DestinationName can be shown to the user. Must be included if DestinationName information is displayed to the user. Not including this parameter or setting a value of 0 indicates that DestinationName is not displayed to the user.
MaxAfterManeuverRoadNameLength	4	uint16	0/1	How many of the widest character in the font used to display AfterManeuverRoadName can be shown to the user. Must be included if After-ManeuverRoadName information is displayed to the user. Not including this parameter or setting a value of 0 indicates that AfterManeuverRoad-Name is not displayed to the user.
MaxManeuverDescriptionLength	5	uint16	0/1	How many of the widest character in the font used to display ManeuverDescription can be shown to the user. Must be included if Maneu-verDescription information is displayed to the user. Not including this parameter or setting a value of 0 indicates that ManeuverDescription is not displayed to the user.
MaxGuidanceManeuverStorageCapacity	6	uint16	0/1	Number of maneuvers the accessory can store at once. Set to 0 to receive all maneuvers for a route.
MaxLaneGuidanceDescriptionLength	7	uint16	0/1	How many of the widest character in the font used to display LaneGuidanceDescription can be shown to the user. Must be included if LaneGuid-anceDescription information is displayed to the user. Not including this parameter or setting a value of 0 indicates that LaneGuidanceDescrip-tion is not displayed to the user.

MaxLaneGuidanceStorageCapacity	8	uint16	0/1	Specifies the number of LaneGuidanceInformation the accessory can store at once. Must be included to receive Lane Guidance instructions. Set to 0 to recieve all Lane Guidance instructions for a route.

15.7.2 IdentificationRejected

Source	ID
Device	0x1D03

Table 15-49: `IdentificationRejected` message parameters

Name	ID	Type	#	Notes
VehicleInformationComponent	20	none	0/1	One or more of the identified Vehicle Information components is not supported by the device
VehicleStatusComponent	21	none	0/1	One or more of the identified Vehicle Status components is not supported by the device
WirelessCarPlayTransportComponent	24	none	0+	One or more of the identified Wireless CarPlay Transport components is not supported by the device
RouteGuidanceDisplayComponent	30	none	0+	One or more of the identified navigation route guidance display components is not supported by the device

15.8 Addendum: Communications

For more information, see "10 Addendum: Communications" (page 193).

15.8.1 StartListUpdates

Source	ID
Accessory	0x4170

Table 15-50: `StartListUpdates` message parameters

Name	ID	Type	#	Notes
RecentsListProperties	1	group	0/1	See Table 15-51 (page 225); Sending this parameter will cause the device to send RecentsListAvailable, RecentsListCount, and RecentsList
RecentsListMax	3	uint16	0/1	Max entries to send; 0 = no limit
RecentsListCombine	4	bool	0/1	Combine calls to the same person; Default = 1
FavoritesListProperties	6	group	0/1	See Table 15-52 (page 226); Sending this parameter will cause the device to send FavoritesListAvailable, FavoritesListCount, and FavoritesList
FavoritesListMax	8	uint16	0/1	Max entries to send; 0 = no limit

Table 15-51: `RecentsListProperties` parameter group

Name	ID	Type	#	Notes
Index	0	none	1	
RemoteID	1	none	1	
DisplayName	2	none	1	
Label	3	none	0/1	
AddressBookID	4	none	0/1	
Service	5	none	1	
Type	6	none	1	
UnixTimestamp	7	none	0/1	
Duration	8	none	0/1	
Occurrences	9	none	1	

Table 15-52: `FavoritesListProperties` parameter group

Name	ID	Type	#	Notes
Index	0	none	1	
RemoteID	1	none	1	
DisplayName	2	none	1	
Label	3	none	0/1	
AddressBookID	4	none	0/1	
Service	5	none	1	

15.8.2 ListUpdate

Device	ID
Device	0x4171

Table 15-53: `ListUpdate` message parameters

Name	ID	Type	#	Notes
RecentsListAvailable	0	bool	0/1	
RecentsList	1	group	0+	See Table 15-54 (page 227)
RecentsListCount	2	uint16	0/1	Number of entries in the list
FavoritesListAvailable	5	bool	0/1	
FavoritesList	6	group	0+	See Table 15-55 (page 227)
FavoritesListCount	7	uint16	0/1	Number of entries in the list

Table 15-54: `RecentsList` parameter group

Name	ID	Type	#	Notes
Index	0	uint16	1	Request index of list entry
RemoteID	1	utf8	1	Remote phone number or email
DisplayName	2	utf8	1	
Label	3	utf8	0/1	Label if in contacts, location if not
AddressBookID	4	utf8	0/1	
Service	5	enum	1	See Table 15-56 (page 227)
Type	6	enum	1	See Table 15-57 (page 228)
UnixTimestamp	7	uint64	0/1	Unix timestamp in UTC
Duration	8	secs32	0/1	Only if occurrences = 1
Occurrences	9	uint8	1	

Table 15-55: `FavoritesList` parameter group

Name	ID	Type	#	Notes
Index	0	uint16	1	Request index of list entry
RemoteID	1	utf8	1	Remote phone number or email
DisplayName	2	utf8	1	
Label	3	utf8	0/1	Label if in contacts, location if not
AddressBookID	4	utf8	0/1	
Service	5	enum	1	See Table 15-56 (page 227)

Table 15-56: `ListUpdateService` enum

Value	Description
0	Unknown
1	Telephony
2	FaceTimeAudio
3	FaceTimeVideo

Table 15-57: `ListUpdateRecentsListType` enum

Value	Description
0	Unknown
1	Incoming
2	Outgoing
3	Missed

15.8.3 StopListUpdates

Source	ID
Accessory	0x4172

Table 15-58: `StopListUpdates` message parameters

Name	ID	Type	#	Notes
This message has no parameters.				

15.9 Addendum: Device Notifications

For more information, see "11 Addendum: Device Notifications" (page 195).

15.9.1 WirelessCarPlayUpdate

Deprecated: WirelessCarPlayUpdate is deprecated.

Source	ID
Device	0x4E0D

Table 15-59: `WirelessCarPlayUpdate` message parameters

Name	ID	Type	#	Notes
Status	0	enum	1	See Table 15-60 (page 229).

Table 15-60: `WirelessCarPlayUpdateStatusenum`

Value	Description
0	Unavailable
1	Available

15.9.2 DeviceTransportIdentifierNotification

Deprecated: DeviceTransportIdentifierNotification is deprecated.

Source	ID
Device	0x4E0E

Table 15-61: `DeviceTransportIdentifierNotification` message parameters

Name	ID	Type	#	Notes
BluetoothTransportIdentifier	0	utf8	0/1	iAP2 Bluetooth transport identifier.
USBTransportIdentifier	1	utf8	0/1	iAP2 USB transport identifier.

15.10 Addendum: Location

For more information, see "12 Addendum: Location Information" (page 196).

15.10.1 GPRMCDataStatusValuesNotification

Source	ID
Device	0xFFF0

Table 15-62: `GPRMCDataStatusValuesNotification` message parameters

Name	ID	Type	#	Notes
GPRMCDataStatusValueA	0	none	0/1	If present, the accessory may report GPRMC Data status field as 'A'.
GPRMCDataStatusValueV	1	none	0/1	If present, the accessory may report GPRMC Data status field as 'V'.
GPRMCDataStatusValueX	2	none	0/1	If present, the accessory may report GPRMC Data status field as 'X'.

Revision History

This chapter describes the changes to *Accessory Interface Specification CarPlay Addendum* since R5.

New Content

Section
"3.2.7.2.9 Auxiliary Input Audio - Speech Recognition" (page 88)
"3.2.7.2.10 Auxiliary Output Audio - Speech Recognition" (page 89)
Table 3-61 (page 117)
Table 3-62 (page 117)
Table 3-78 (page 126)
Table 3-79 (page 127)
Table 3-80 (page 127)
Table 3.3.5.4.3 (page 151)
Table 3-130 (page 177)
Table 15-22 (page 211)

Updated Content

Section	Change Summary
"3.2.2.4 Speakers and Microphone" (page 21)	Updates for Siri activation.
"3.2.2.5 Siri Button" (page 21)	Updates for Siri activation.
"3.2.3 Architecture" (page 24)	AAC-ELD removed from 'Topology of an automotive head unit using CarPlay'.
"3.2.5.1.5 Security" (page 37)	Updates to 'Wi-Fi Protected Access (WPA) Modes' related to CarPlayStartSession message.
"3.2.5.1.8 Wi-Fi Alliance Compliance and Conformance" (page 42)	Updates for MIMO (2x2) configurations.
"3.2.6.4 iAP2 Client over CarPlay Client" (page 61)	Update to AccessoryIdentification requirement.
"3.2.7.1.2 Instrument Cluster Display" (page 64)	Updates for optional map UI elements, zoom level controls and timing requirements for the UI to appear.

"3.2.7.1.5 Safe Area" (page 68)	Updates related to drawing CarPlay UI outside of a safe area on the main display.
"3.2.7.2 Audio" (page 78)	Updates related to auxiliary audio and removing requirements for AAC-ELD.
"3.2.7.2.2 Main Audio - Telephony" (page 80)	Removed requirements for AAC-ELD.
"3.2.7.2.4 Main Audio - Speech Recognition" (page 84)	Renamed from 'Siri' to 'Speech Recognition'.
"3.2.7.2.11 Mixing" (page 90)	Updates to include auxiliary audio.
"3.2.7.2.12 Volume Management" (page 90)	Updates to include auxiliary audio.
"3.2.7.2.13 Ducking" (page 91)	Updates to include auxiliary audio.
Table 3-32 (page 100)	Updates to nightMode and added enhancedSiriInfo.
Table 3-34 (page 105)	Deprecated AAC-ELD audio formats.
Table 3-38 (page 110)	Added drawUIOutsideSafeArea.
Table 3-64 (page 118)	Added "enhancedSiri".
Table 3-65 (page 119)	Updated 'Required?' conditions.
Table 3-66 (page 119)	Added "enhancedSiri".
Table 3-67 (page 120)	Removed audioLoopback, added streamStartTimestamp and burstPeriodMs.
Table 3-72 (page 123)	Added Aux Out Audio and Aux In Audio.
Table 3-73 (page 123)	Added Aux Out Audio and Aux In Audio to speechRecognition.
Table 3-74 (page 124)	Added RTP Auxiliary Audio Input and RTP Auxiliary Audio Output.
Table 3-77 (page 126)	Added SET_PARAMETER.
"3.3.3.1 Resources" (page 128)	Updates related to Siri and auxiliary audio.
"3.3.5 User Input" (page 136)	Updates for map zoom level controls for instrument cluster displays.
"3.3.7 Activating Siri" (page 157)	Updates to support instant button and voice activation for Siri.
"3.3.8.6 forceKeyFrame" (page 166)	Added uuid key.
"3.3.8.9 requestSiri" (page 167)	Updates to requestSiri keys and Siri actions enum values.
"3.3.8.23 showUI" (page 174)	Added support for optional map UI elements.

"3.3.8.27 mapAppearanceUpdate" (page 176)	Corrected descriptions for appearanceMode and appearanceSetting.
"6 Route Guidance" (page 183)	Added requirement related to built-in navigation system metadata support.
"15.1.2 AccessoryWiFiConfigurationInformation" (page 199)	Updates related to channel width protocols and AccessoryWiFiConfigurationSecurityType enum.
Table 15-15 (page 207)	RouteGuidanceDisplayComponentID will always be included in RouteGuidanceManeuverInformation.
Table 15-38 (page 218)	Updates related to channel width protocols, IPv6 zone indexes and CarPlayStartSession Wi-Fi SecurityType enum.
Table 15-46 (page 222)	Updates to clarify when RangeWarning parameters should be used.

Apple Inc.
One Apple Park Way
Cupertino, CA 95014
408-996-1010