

# Head Unit Integration Guide

Version 2.4.0, March 29, 2019

The Head Unit Integration Guide (HUIG) describes how automotive in-vehicle infotainment (IVI) systems can leverage the compute power, connectivity, and applications of modern mobile devices through Android Auto Projection (AAP). The guide is intended for automotive IVI development teams who want to integrate AAP on a head unit.

This version of the HUIG (2.4.0) is associated with the following components:

Component	Version
<a href="#">AAP Protocol Specification</a>	1.5
<a href="#">AA Partner Development Kit (PDK)</a>	1.5
<a href="#">GAL Receiver Library API Reference</a>	1.5

## Table of Contents

- [About this document](#)
- [0. Introduction](#)
  - [0.1. Conventions](#)
  - [0.2. About Android Auto Projection \(AAP\)](#)
  - [0.3. Transport](#)
  - [0.4. Channels](#)
  - [0.5. Head Unit components & requirements](#)
  - [0.6. Vehicle types](#)
- [1. Icons](#)
  - [1.1. Android Auto Launch Icon](#)
  - [1.2. Android Auto Label Icon](#)
  - [1.3. Launch Icon examples](#)
  - [1.4 Icon Usage](#)
- [2. Receiver Library](#)
  - [2.1. Receiver Library details](#)
- [3. USB connection](#)
  - [3.1. Setting up transport](#)
  - [3.2. USB connection monitoring](#)
  - [3.3. Handling disconnect messages](#)
- [4. Connection and launch process](#)
  - [4.1. Connect AAP](#)
  - [4.2. Authentication](#)
  - [4.3. Service Discovery Request](#)
  - [4.4. Service Discovery Response](#)
  - [4.5. First run experience](#)
  - [4.6. Manage applications](#)
  - [4.7. Initial video focus](#)
  - [4.8. Run tutorial](#)

- 4.9. Start Android Auto UI
- 4.10. Disconnecting AAP
- 4.11. Handling concurrent connections
- 4.12. Authentication certificates
- 4.13. Authentication sequence
- 5. Wireless connection
  - 5.1. Wireless hardware
  - 5.2. Wireless connection process overview
  - 5.3. Bluetooth service discovery
  - 5.4. Bluetooth RFCOMM control channel
  - 5.5. TCP connection
  - 5.6. Starting wireless projection
  - 5.7. Wireless projection error handling
  - 5.8. Switching projection sources
- 6. Bluetooth
  - 6.1. Bluetooth service announcement
  - 6.2. Protocol messages for Bluetooth pairing
  - 6.3. Bluetooth pairing scenarios
  - 6.4. HFP connection
  - 6.5. Bluetooth audio (A2DP) connection
  - 6.6. Bluetooth reconnection
  - 6.7. Phone calls from other HFP devices or embedded SIMs
  - 6.8. Clarifications
  - 6.9. Additional Bluetooth profiles
  - 6.10. Bluetooth service announcement
  - 6.11. Bluetooth pairing and connection
- 7. Video
  - 7.1. Integration types
  - 7.2. Video decoder
  - 7.3. Frame rate
  - 7.4. Display area and screen
  - 7.5. Video resolution and setup
  - 7.6. UI resolution
  - 7.7. Video characteristics
  - 7.8. Video latency and flow control
  - 7.9. Android Auto UI stream types
  - 7.10. Video focus
  - 7.11. Competing functions
- 8. Audio
  - 8.1. Terminology
  - 8.2. Audio streams
  - 8.3. Audio focus
  - 8.4. Audio latency and flow control
  - 8.5. Audio volume
  - 8.6. Audio focus examples
- 9. Input devices
  - 9.1. Touchscreen

- 9.2. Touchpad
- 9.3. Rotary input
- 9.4. Hardware input actions
- 9.5. Software input actions
- 9.6. Handling of input actions
- 9.7. Source input
- 9.8. Keycodes
- 10. Automatic Speech Recognition
  - 10.1. Native ASR
  - 10.2. Initiating ASR
  - 10.3. ASR and reverse backup camera
  - 10.4. Speech audio format
  - 10.5. Branding ASR
  - 10.6. ASR session notifications
  - 10.7. ASR short-press implementation
  - 10.8. ASR long-press implementation
- 11. Navigation
  - 11.1. Navigation focus request
  - 11.2. Navigation focus notification
  - 11.3. Integrating navigation
- 12. Vehicle data
  - 12.1. Navigation data
  - 12.2. Other Vehicle data
  - 12.3. Driving Status bitmask definition
- 13. Application status and data
  - 13.1. Navigation data
  - 13.2. Accessing media data
  - 13.3. Accessing Telephony data
- 14. Vendor Extension Channels (VECs)
  - 14.1. VEC architecture
  - 14.2. Implementing the VEC
  - 14.3. VEC sample HU implementation
- 15. Radio
- Appendix A: Release Notes
  - HUIG 2.4
  - HUIG 2.3.1
  - HUIG 2.3
  - HUIG 2.2.x
  - HUIG 2.1
- Appendix B: Requirements Change Log
  - HUIG 2.4
  - HUIG 2.3.1
  - HUIG 2.3
  - HUIG 2.2.x
  - HUIG 2.1

Today's mobile devices are packed with technology - the latest hardware and software, high-speed mobile networks, and an ever-expanding universe of connected services and applications. Their convergence empowers users with new ways to communicate, enjoy digital content, access information, learn, and navigate the world around them. Unfortunately these devices were not designed to be used while driving.

Android Auto ([www.android.com/auto](http://www.android.com/auto)) enables seamless integration between mobile devices and vehicles, allowing drivers to access and control technology on mobile devices using an interface that's easier and safer to use while driving.

This document describes the integration that enables automotive in-vehicle infotainment (IVI) systems to leverage the compute power, connectivity, and applications of modern mobile devices through Android Auto Projection (AAP) technology. It is intended for automotive infotainment development teams who want to integrate AAP on a head unit. It includes the required and recommended human interactions to deliver a predictable and high quality user experience.

This document does not describe the low-level AAP protocol implementation abstracted by a sender and receiver library on both the head unit (HU) and the mobile device (MD). For these details, refer to the [AAP Receiver Library documentation](#) and the [AAP Protocol Specification](#).

## 0.1. Conventions

The use of "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" is per the IETF standard defined in [RFC2119](#)

## 0.2. About Android Auto Projection (AAP)

AAP bridges mobile devices and vehicles to enable drivers to access the capabilities of the device through the physical human machine interface (HMI) controls provided by the vehicle. The MD manages all user interface (UI), software logic, connectivity, and compute power for applications, and projects these applications into the vehicle through AAP.

AAP architecture leverages the strongest attributes of both vehicle and mobile device to create an optimal driving experience.

From the vehicle HU, the Android Auto integration leverages:

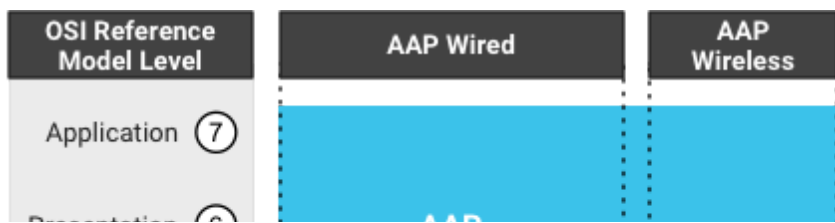
- Human machine interface (HMI) better suited for use while driving.
- Highly accurate vehicle data with better information about the driving environment.

From the MD, the Android Auto integration leverages:

- Latest computing hardware platform
- Latest software platform
- Latest high speed data connectivity to the cloud via latest mobile data networks
- Evolution of hardware, software, and connectivity at consumer electronics pace

## 0.3. Transport

The AAP protocol itself is transport-agnostic, and runs over any transport with sufficient bandwidth. The platform implementation supports USB 2.0 (since inception) and Wireless Local Area Network (Wi-Fi) with the HU as the access point (since v1.4).



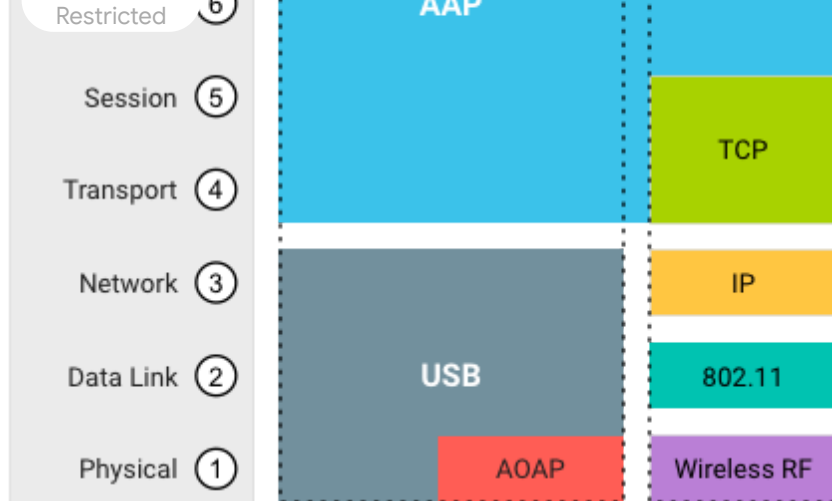


Figure 0.1. Android Auto network stack

## 0.4. Channels

The AAP protocol includes separate channels to manage data streams between MD and vehicle:

- **Control** (*bi-directional*). Manages link initialization and setup of channels for media, input, etc.
- **Video output** (*device-to-vehicle*). Sends H.264 video from the MD to the vehicle HU for display on the main console display.
- **Audio output** (*device-to-vehicle*). Carries audio from the MD to the vehicle HU for output through the vehicle speakers.
- **Vehicle data** (*vehicle-to-device*). Carries vehicle-associated (GPS, wheel speed, etc.) data from the vehicle to the MD.
- **Input** (*vehicle-to-device*). Sends input events from input devices on the vehicle HU (such as touchscreens, buttons, controllers, etc.) to the MD.
- **Microphone audio** (*vehicle-to-device*). Used by the MD to receive audio captured by vehicle microphone(s).
- **Bluetooth HFP** (*bi-directional*). Used for phone call audio.
- **Application status and data** (*device-to-vehicle*). Navigation status (turn by turn metadata) and media playback status.

## 0.5. Head Unit components & requirements

The head unit architecture consists of the following high-level components unique to AAP:

- **HU OS**. The operating system (Android, Linux, QNX, Windows, etc.) environment software stack that powers the on-board infotainment experience.
- **OS adaptation layer**. The OS-specific abstraction layer between the HU OS and the AAP receiver. The layer implements AAP receiver bindings and translates the message call and data to native HU OS subsystems. Google provides reference implementations for Android and QNX in source form, but does not provide a commercial implementation.
- **Android Auto Projection Receiver**. The HU implementation of AAP that connects with HU OS interfaces (video, audio, connectivity, input, etc.) through the OS Adaptation Layer to enable an integrated user experience. Google provides the Receiver Library as portable C++ source code.

**NOTE:** This document does not provide a detailed overview of the entire HU software stack.

## 0.6. Vehicle types

AAP MAY be integrated into HUs shipped in cars (R00-010). AAP MAY be integrated into aftermarket HUs designed for cars (R00-020). The term *car* in this document is used to refer to a vehicle that does not require a commercial

dl. Restricted to operate. AAP MAY be integrated into HU shipped in trucks (R00-030). The term *truck* in this document is used to refer to a vehicle that requires a commercial driver's license to operate.

Navigation directions provided in Android Auto by Google Maps and other navigation applications are not designed for use by trucks. If AAP is integrated into a truck, the HU MUST NOT grant navigation focus to the MD when requested (R00-040). When an HU in a truck rejects a navigation focus request from the MD, it MUST display a transient native overlay with the following text (or a region-appropriate translation of this text): "Android Auto navigation is not available in this vehicle" (R00-050). If AAP is integrated into a truck, when starting AAP, the HU MUST display a dismissable native disclaimer with the following text (or a region-appropriate translation of this text): "Android Auto navigation apps are not designed for commercial vehicles and may not be used for navigation" (R00-060). An HU MAY suppress this start-up disclaimer for the second or subsequent connections of a specific MD if the HU receives and remembers per-MD specific user intent to hide this message (through user selection of a 'Do Not Show Again' option, etc.)(R00-070).

## 1. Icons

Android Auto specifies two icons: a Launch Icon (color and monochrome) and a Label Icon (monochrome only).

The color and monochrome icons (vector files) can be found here in the [Android Auto Partner Help Center](#).

### 1.1. Android Auto Launch Icon

Android Auto provides two Launch Icon options: Color or Monochrome. The HU MUST use a Launch Icon consistent with its native icon style (R01-020).

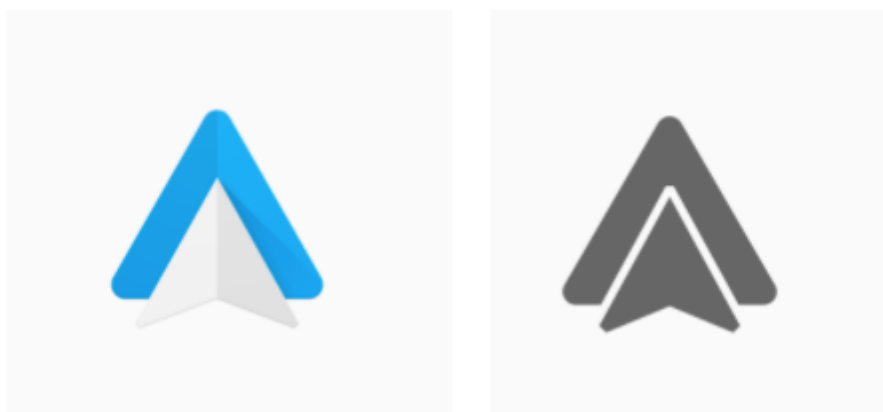


Figure 1.1. Android Auto color Launch Icon and monochrome Launch Icon.

#### 1.1.1. Launch Icon states

The Launch Icon has three states: Unavailable, Available, and Selected. The HU MUST show the Launch Icon in the available state when an AAP session is active (R01-030). A wireless HU MUST also show the Launch Icon in the available state when an MD is available to start wireless projection (Bluetooth RFCOMM connection and version negotiation are complete) (R01-035). For details on wireless projection Bluetooth RFCOMM connection, refer to [Bluetooth RFCOMM control channel](#).

HUs distributed in countries where AAP has launched SHOULD show the Launch Icon in the unavailable state when an MD is not connected or a previous AAP session has been exited by the user (R01-040). HUs distributed in countries where AAP has not launched MUST NOT show the Launch Icon unless an AAP session is active or an MD is available to start wireless projection (R01-050).

When the user presses or selects the Launch Icon, the HU SHOULD render it in the Selected state (R01-060).



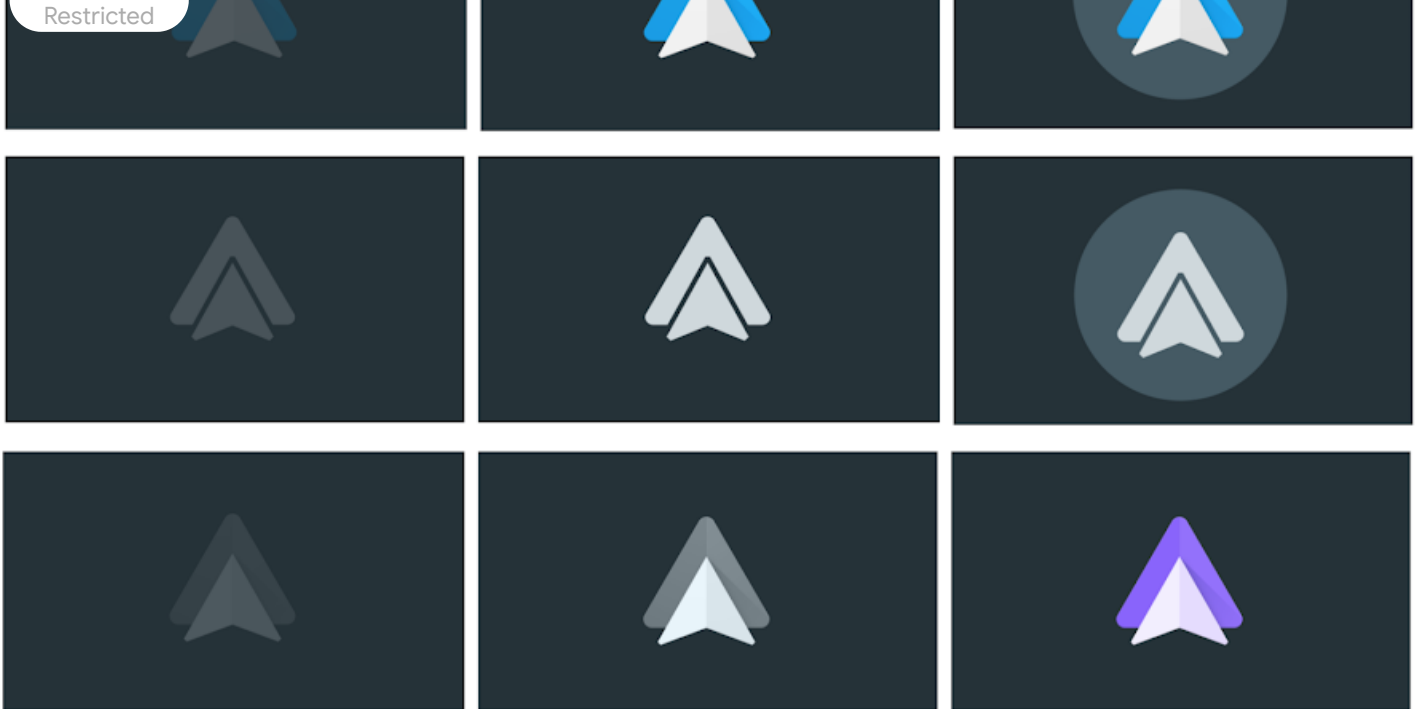


Figure 1.2. Android Auto Launch Icon states.

### 1.1.2. Launch Icon labelling

The HU SHOULD display a text label next to the Android Auto Launch Icon in a manner consistent with other native icons (R01-070). If the HU displays a text label next to the Android Auto Launch Icon, that label MUST be the text string "Android Auto" (R01-080).

### 1.1.3. Launch Icon placement

Native buttons or affordances that launch Android Auto MUST use the Android Auto Launch Icon (R01-085).

An AAP session is active when the MD is sending the projection video stream to the HU over USB or Wi-Fi irrespective of video focus.

When an AAP session is inactive, the Launch Icon MUST be placed on the home screen in a location that enables the user to launch Android Auto in no more than two (2) gestures (R01-090). For example, the Launch Icon may be placed on the home screen of a rotary-enabled HU and Android Auto launched with one rotate gesture followed by a select action, or placed on the second page of a touch-enabled HU, and Android Auto launched with one swipe gesture followed by one tap.

When an AAP session is active, the Launch Icon MUST be placed directly on the home screen to enable the user to launch Android Auto in one gesture (R01-095).

### 1.1.4. Launch Icon colorization

The Launch Icon MAY be colorized to conform to the patterns for native system icons by customizing the outer-top chevron of the Launch Icon (R01-100). The foreground/smaller chevron of the Launch Icon MUST remain white (R01-110). The Launch Icon MUST NOT be made transparent (R01-115).

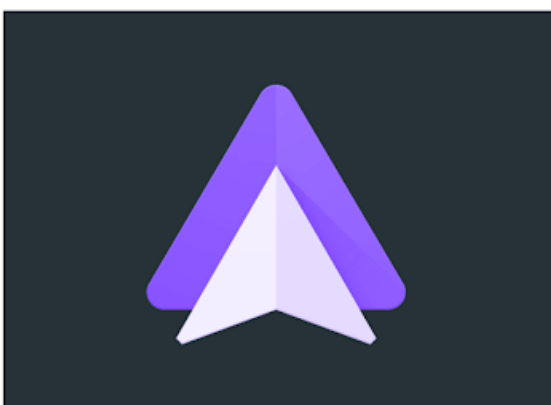


Figure 1.3. Launch Icon colorization examples.

If the Launch Icon is shown in a colored container, the Launch Icon MUST be colored white (R01-120). If the native OEM system does not require a specific color for the Launch Icon container, then light blue 500 (#03A9F4) from the Material Design color specification MUST be used (R01-130).



Do.



Don't.

Figure 1.4. Launch Icon container examples.

### 1.1.5. Launch Icon actuation

When a user actuates (taps or selects) the Launch Icon:

- If a projection session is active, the HU MUST grant video focus to the MD (R01-131).
- If a projection session is not active, and the HU does not support wireless projection, the HU MUST display a native message informing the user to connect an Android phone to the car with a USB cable to start using Android Auto (R01-132).
- If a projection session is not active, the HU does support wireless projection and no MDs are available for wireless projection, the HU MUST display a native message informing the user to connect an Android phone to the car with a USB cable or to pair an Android phone via Bluetooth to start using Android Auto (R01-133).
- If a projection session is not active, the HU does support wireless projection and one MD is available for wireless projection, the HU MUST send a [WifiStartRequest](#) to that MD (R01-134).
- If a projection session is not active, the HU does support wireless projection and multiple MDs are available for wireless projection, the HU MUST show a native screen listing all MDs available for wireless projection (R01-135).

## 1.2. Android Auto Label Icon

Native widgets or screens in the HU MUST show some form of Android Auto attribution on all content originating from or pertaining to Android Auto (R01-140). This attribution MUST be displayed as an Android Auto Label icon, an Android Auto text label, OR a combination of the Android Auto Label icon and text label (R01-141).

If an icon is used, the HU SHOULD use the monochrome Label Icon but MAY use an icon with a gray-colored outside chevron as a label icon instead of monochrome (R01-143). If a text label is used, that label MUST be the text string "Android Auto" (R01-145). For requirements on use of the Android Auto Label Icon when displaying Android Auto media metadata in the cluster, see [Accessing media data](#).







Figure 1.5. Monochrome Android Auto Label Icon.



Figure 1.6. Alternative Android Auto Label Icon.

### 1.3. Launch Icon examples

The following images display examples of permitted Android Auto Launch Icon treatments.

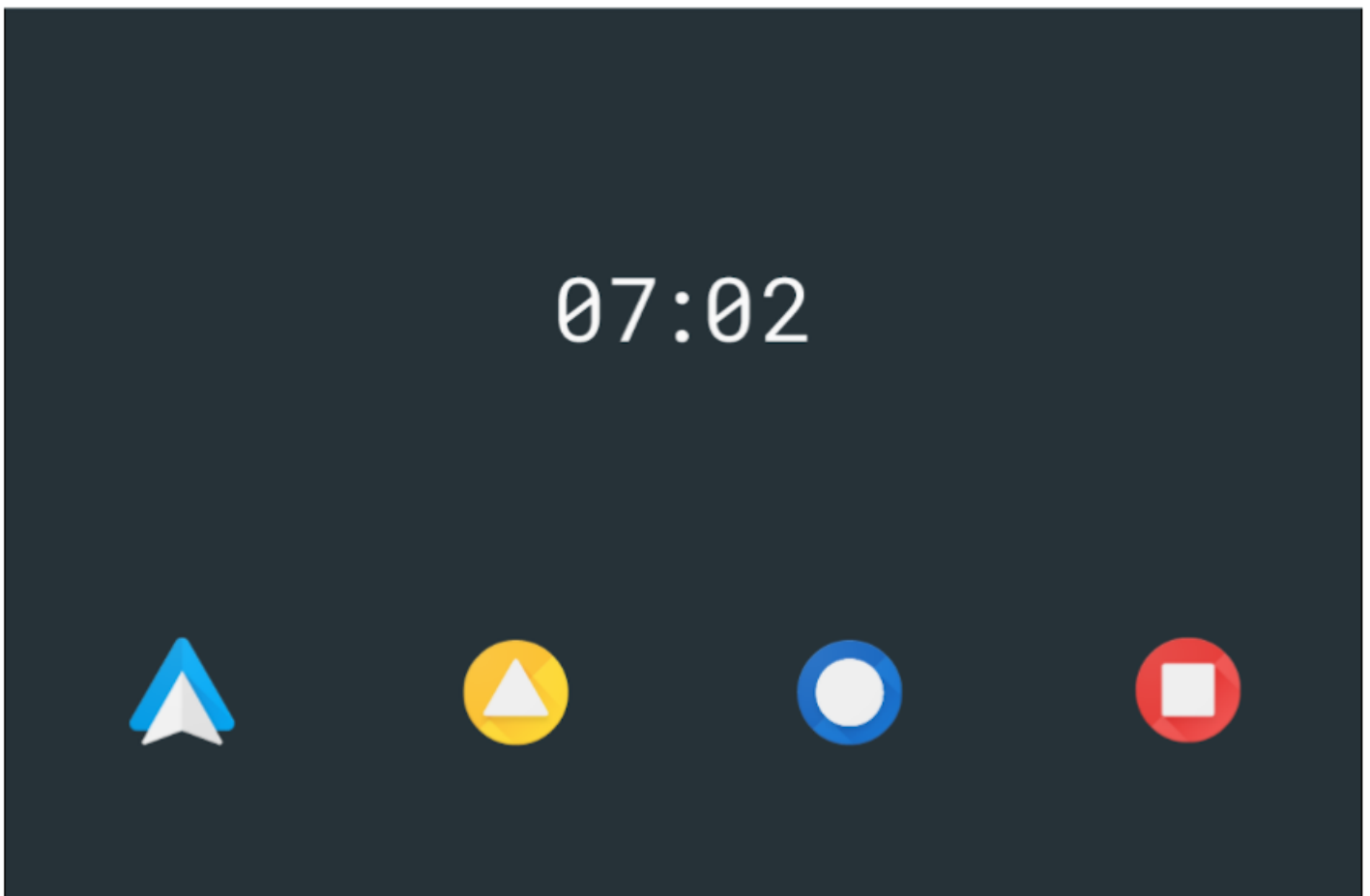
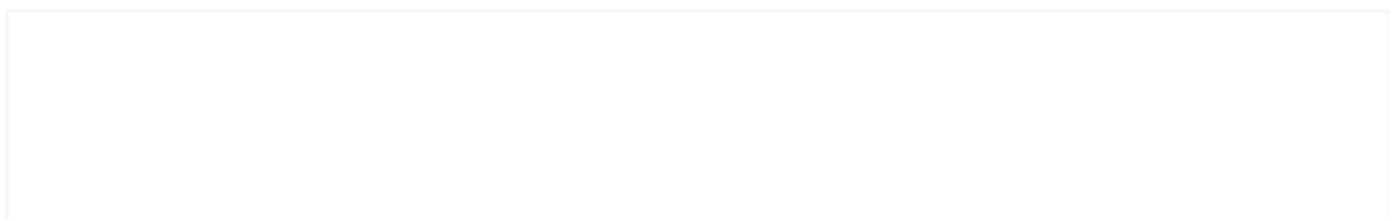


Figure 1.6. 3D style iconography.



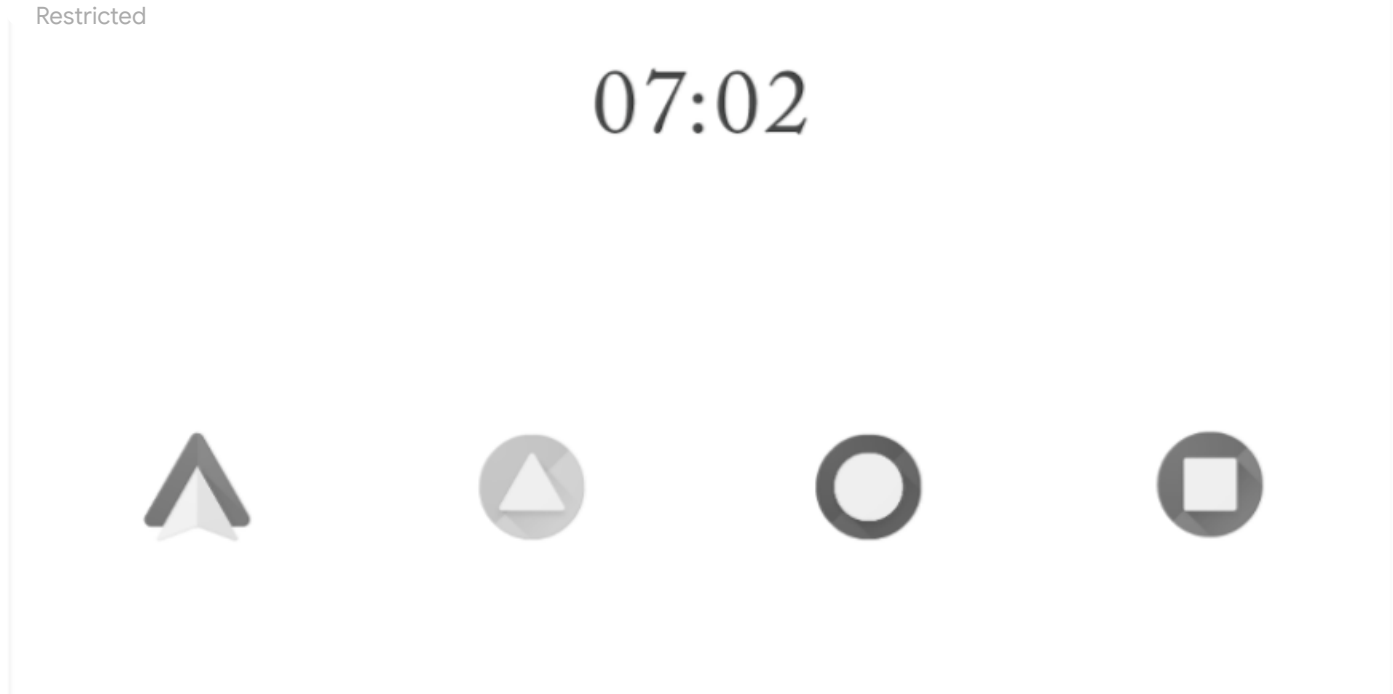


Figure 1.7. Monochromatic 3D style iconography.

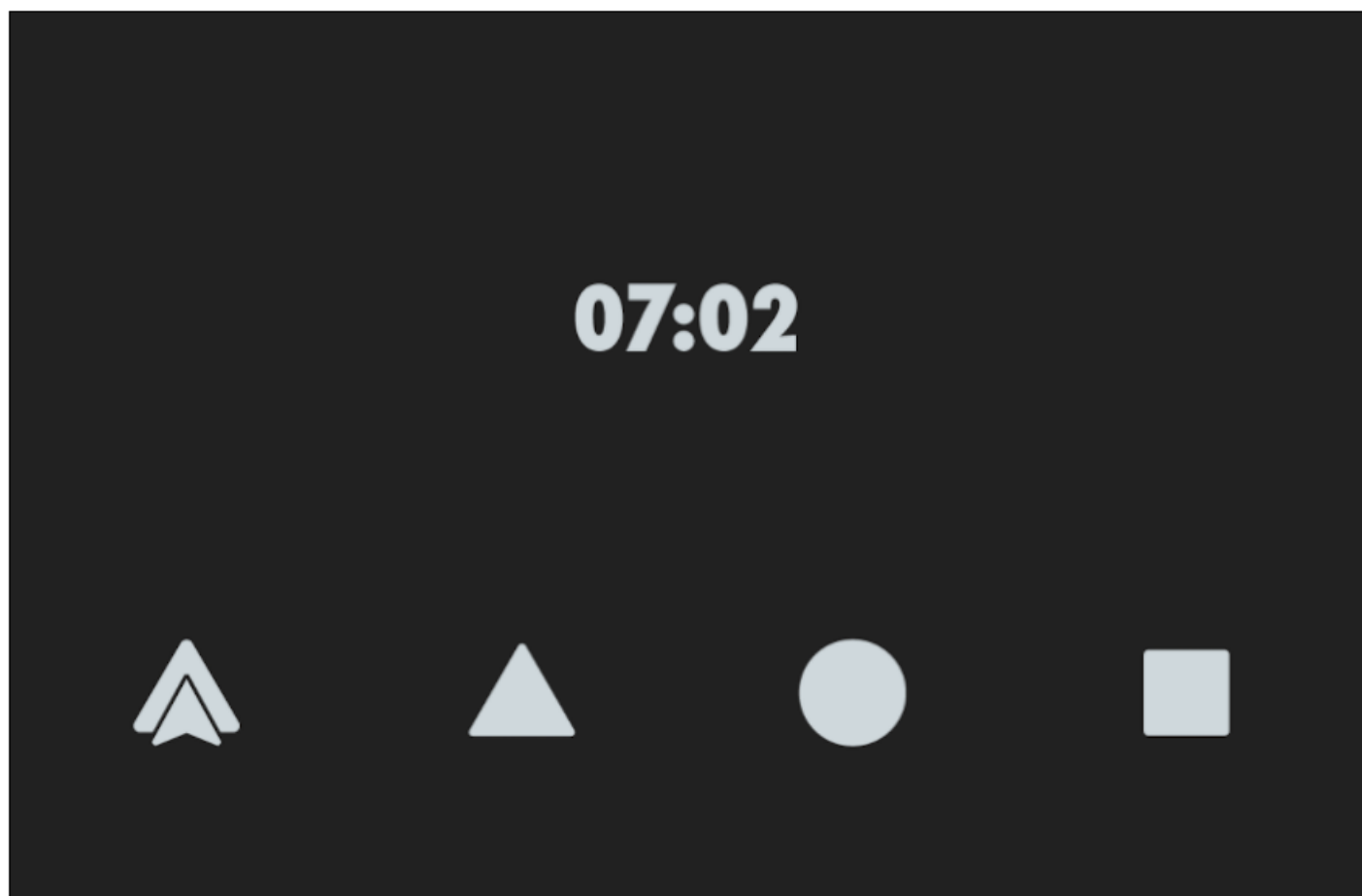


Figure 1.8. Glyph style iconography.

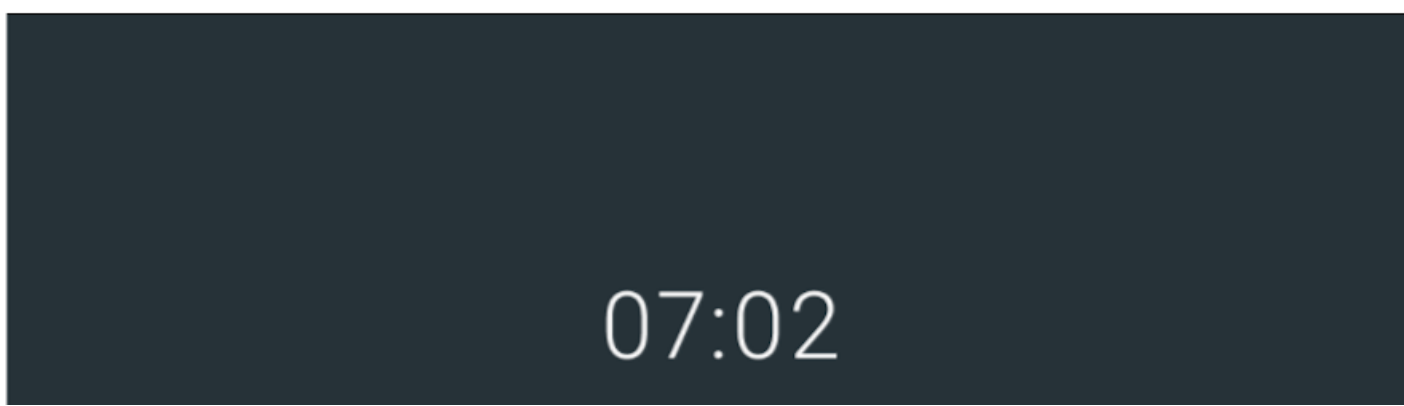




Figure 1.9. Framed glyph style iconography.

## 1.4 Icon Usage

If Android Auto icons are resized to match other icons in the native UI, the resizing **MUST** be performed proportionally (R01-150). Android Auto icons **MUST NOT** be resized smaller than 8mm (R01-160). Android Auto icons **MUST NOT** overlay native content (R01-170). Android Auto icons **MUST NOT** be made transparent (R01-180).

## 2. Receiver Library

The AAP Receiver Library provides a full protocol endpoint solution in the form of a portable C++ library for the HU. The integrator's build environment **MUST** support **C++11** to be able to integrate the Receiver Library releases in 2018 (R02-003). Integrators should expect that **C++17** support will be required from 2019.

The OS Adaptation Layer connects the AAP receiver and the host OS for end-to-end integration. To act as an endpoint, an HU **MUST** include the AAP Receiver Library and implement the operating system (OS) Adaptation Layer that associates Receiver Library APIs to the native HU OS (R02-010).

The Receiver Library implements and encapsulates AAP interactions between the HU and MD. To ensure compatibility with MD implementations, integrators **MUST NOT** modify or re-implement this protocol (R02-020).

The OS Adaptation Layer **MUST** support C++ STL and Atomic Refcounting (R02-030).

### 2.1. Receiver Library details

The Receiver Library handles the low-level protocol packet-framing, channel management, authentication, etc. However, it does not include OS-specific bindings required to complete the integration with relevant (video, audio, input, etc.) HU subsystems. Google provides reference implementations of the OS Adaptation Layer for Android, Linux, and QNX-based systems. Integrators are responsible for the commercial implementation of the OS adaptation layer, but may use these reference implementations as a starting point.

In the source code distribution, the library is compiled as part of the Android build. Google also includes a standalone build file to compile the Receiver Library independently from the Android build system. The Receiver Library may need to be recompiled with the platform's build toolchain for compatibility with the HU OS environment. The makefile may need to be updated to compile the Receiver Library with the same build tools and system libraries as required for the HU software build.

Open Source code is used by the AAP receiver in the following components:

- Protocol Buffers ([code.google.com/p/protobuf/](https://code.google.com/p/protobuf/)) uses a new BSD license.
- OpenSSL ([www.openssl.org](http://www.openssl.org)) uses a BSD-like license (not included in repo but linked).

The base Receiver Library uses the following C++ STL classes:

• `std::deque`

- `Std::set`
- `Std::string`
- `std::vector`

The Receiver Library uses atomic reference counting for memory management and requires access to an atomic *increment and return* and atomic *decrement and return*. The supplied implementation utilizes the GNU Compiler Collection (GCC) builtin `__sync_add_and_fetch()`.

### 2.1.1. Interfaces

The majority of AAP integration work involves implementing receiver interfaces. The document provides an overview of the integration architecture, message flows, and interface definitions. This document does not provide details for interface API definitions, method signatures, and parameters. For these details, refer to the [AAP Receiver Library documentation](#).

### 2.1.2. Version compatibility

Future versions of the AAP protocol may introduce new features supported only on MDs and HUs that implement the new AAP version. However, future AAP versions will maintain backwards compatibility with older versions wherever possible. If maintaining protocol backwards compatibility is not possible and an MD is incapable of running an earlier AAP version, the device will notify the user of the compatibility issue upon initial connection to the HU and immediately terminate the AAP connection.

## 3. USB connection

An AAP connection begins immediately when a user performs one of the following actions:

- Plugs the MD into the vehicle USB connection using a standard USB connector
- Powers on the HU with the MD already plugged in

Silently probing the HU using USB is currently a beta feature (R03-200-B); therefore, the HU SHOULD not use the `GalReceiver::setProbeOnlySession()` flag. After an MD is connected to the vehicle, the HU initializes the USB host as an Android Open Accessory (AOA) device. The HU MUST be a USB 2.0 Hi-Speed Host or USB 3.0 Host and provide host support for the [Android Open Accessory \(AOA\) protocol V1.0](#) (R03-010). The HU MUST NOT support AOA V2.0 (R03-015). The vehicle or HU:

- MUST provide at least one (1) standard USB-A port or USB-C port that supports AAP (R03-020)
- MAY provide multiple ports that support AAP (R03-030)
- MUST clearly label ports that support AAP if the vehicle or HU has any ports that don't support AAP (R03-040)

All USB ports that support AAP:

- MUST support at least 50 Mbps/s throughput (R03-050)
- MUST support the [USB 2.0 CDP specification](#) (R03-060)
- MUST support USB Hubs (may be exposed in HU debug mode only) (R03-070)
- MUST supply voltage between 4.75V and 5.25V (R03-080)
- MUST be tested to USB2 compliance requirements at the user facing interface including the full vehicle harness in circuit (R03-090)

Validation at the HU interface or with a test harness is not acceptable.

The USB-IF Battery Charging Specification v1.2 section 4.1.4 Shutdown Operation lists three (3) methods of shut down. The HU SHOULD equip USB current-limiting and adequate power supply to avoid VBUS shutdown as this also terminates the AAP connection (R03-100).

### 3.1. Setting up transport

The AAP protocol requires a transport connecting the MD and HU

### 3.1.1. Universal Serial Bus (USB)

HU implementations MUST support USB as a transport for AAP (R03-110). In this configuration, the AAP protocol runs on top of the [Android Open Accessory Protocol](#) (AOAP) to communicate over USB bulk endpoints established between the MD and HU.

### 3.1.2. Switching MD to AOAP mode

MDs and HUs can support multiple USB modes including charge-only, USB Mass Storage, MTP, and AOAP. To ensure compatibility with all Android devices, the HU MUST attempt AOAP initialization regardless of the USB Vendor ID (VID), Product ID (PID), and USB Class of the connected MD (R03-120). To correctly detect and negotiate USB mode selection, the host (HU) and client (MD) use the following process:

1. HU attempts to connect to MD using AOAP and advertises itself as an AOAP accessory.
2. If MD:
  - DOES support AOAP, it successfully re-enumerates in AOAP mode.
  - DOES NOT support AOAP, it ignores the AOAP handshake, enabling the HU to negotiate other USB modes or charge-only mode.

An Android MD identifies a vehicle as a AAP receiver based on accessory identifiers:

Identifier	Value
Manufacturer Name	Android
Model Name	Android Auto
Description	Android Auto
AAP Protocol Version	1.0
URI	
Serial Number	

The [URI](#) and [Serial Number](#) fields are required for the AOAP handshake, but the HU can set them to any value, including the empty string. They are only used to give information to the user if the MD fails to launch Android Auto.

Newer mobile devices might advertise as AOAP 2.0, which is backward-compatible with AOAP 1.0. For details on configuring an AOAP host accessory, refer to [Android Open Accessory Protocol](#).

When the MD is in accessory mode, the vendor ID is 0x18D1 (Google) and the product ID is generally 0x2D00 or 0x2D01. Some MDs do not correctly implement the AOAP specification and may set a Product ID of 0x2D04 and 0x2D05 (which should be presented by the MD only in response to the host enabling AOA audio capability). The HU MUST accept any of the MD PIDs associated with the AOAP accessory capability (0x2D00, 0x2D01, 0x2D04, or 0x2D05) (R03-130).

## 3.2. USB connection monitoring

HUs MUST monitor that USB connection is alive by performing at least one packet read from the connection every second (R03-140). The HU MUST send a [PingRequest](#) message to the MD every second (R03-150). If no response to the Ping message is received within three (3) seconds, the HU MUST show a message to the user informing the user that AAP has stopped, and there is a problem with the connection (R03-160).

### 3.3. Handling disconnect messages

During an active AAP session, the HU MUST listen for a MD *disconnect message* (ByeBye) that indicates the user has exited or disabled AAP (R03-170). If the disconnect message is received, the HU re-enumerates USB and MUST NOT automatically restart AAP (R03-177). A **ByeByeRequest** from the MD includes a **ByeByeReason** and the HU:

- MUST NOT allow the user to start AAP unless the MD is unplugged/replugged if the **ByeByeReason** is 'NOT\_SUPPORTED' or 'NOT\_CURRENTLY\_SUPPORTED' (R03-180).
- MUST allow the user to start AAP without unplugging/replugging the MD if the **ByeByeReason** is 'USER\_SELECTION', 'DEVICE\_SWITCH' or 'PROBE\_SUPPORTED' (R03-185).
- MUST NOT mark an MD as AAP incompatible for future connections unless the **ByeByeReason** is 'NOT\_SUPPORTED' (R03-187).

If an HU that supports wireless AAP has completed RFCOMM version negotiation with an MD, but has never had a wireless session with that MD, after concluding a wired session, the HU SHOULD display native guidance to the user that they can connect wirelessly with their MD the next time (R03-190).

## 4. Connection and launch process

The HU must have AAP enabled and MUST NOT require a user to go into the native settings menu to enable AAP (R04-010). When an Android Auto-capable device is connected via cable (USB projection) or RFCOMM (wireless projection), the HU MUST notify the user that Android Auto is available even when another projection technology is active (R04-415). If the HU supports multiple smartphone integration technologies that work via USB or Wi-Fi with Android phones, the HU MAY display a screen to enable the user to choose between Android Auto and other available smartphone integration technologies (R04-418).

When the user initiates a connection, the MD executes a series of steps to prepare the connection with the vehicle HU. If a step is not necessary, the device skips that step (e.g. if the required applications are already installed and set up, the Manage Applications screen does not appear). Similarly, if a user has previously authorized a step, the device executes that step silently without user interaction.

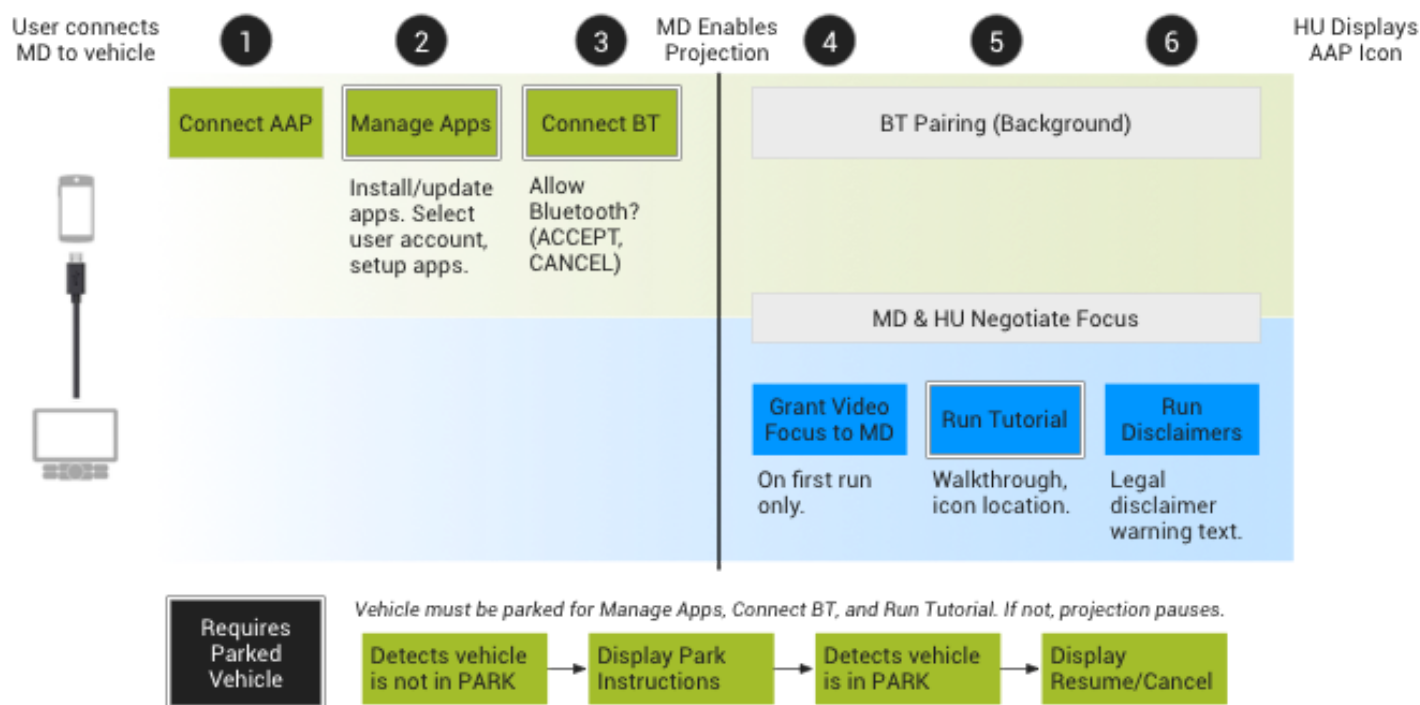


Figure 4.1 Overview of the connection process.

### 4.1. Connect AAP

After establishing an AAP connection (wired projection) or TCP connection (wireless projection) with the MD, the HU initiates negotiation of the AAP connection by sending a Version Request to the MD with the ReceiverLib function **GalReceiver::start()**. The HU MUST NOT send additional Version Requests or ping requests (**GalReceiver::sendPingRequest()**) while awaiting a response from the MD (R04-020). Encrypted packets

(<sup>Restricted</sup> `sendInRequest()`) while awaiting a response from the MD (R04-020). Encrypted packets MUST NOT be sent from the HU to the MD until authentication is complete (R04-030).

## 4.2. Authentication

The AAP sender (on the MD) and the AAP Receiver Library (on the HU) perform a TLS handshake at the beginning of the Service Discovery process. The HU needs a TLS certificate to establish a secure connection.

The HU MUST securely store the certificate and pass it to the Receiver Library on request (R04-040). The HU SHOULD use hardware-backed secure storage (R04-050). The HU MAY use software-backed security (such as Linux file system permission control) (R04-060). To confirm the validity of the HU certificate, the certificate issuing authority and signature are verified, a process that takes place independent of certificate validity dates.

## 4.3. Service Discovery Request

If the MD supports AAP, authentication succeeds and the HU receives a Service Discovery Request, i.e. `ReceiverLib IControllerCallbacks::serviceDiscoveryRequestCallback` is called. The HU MUST inform the user that Android Auto is available and enable the user to start a new Android Auto connection unless there is another AAP session active with another MD (R04-480).

If the MD does not support AAP, then a ByeBye message MAY be received by the HU. Some MDs that do not support AAP will not send a Service Discovery Request *or* a ByeBye message; in this case the HU SHOULD reset the USB connection no less than 15 seconds after calling `GalReceiver::start()` (R04-070).

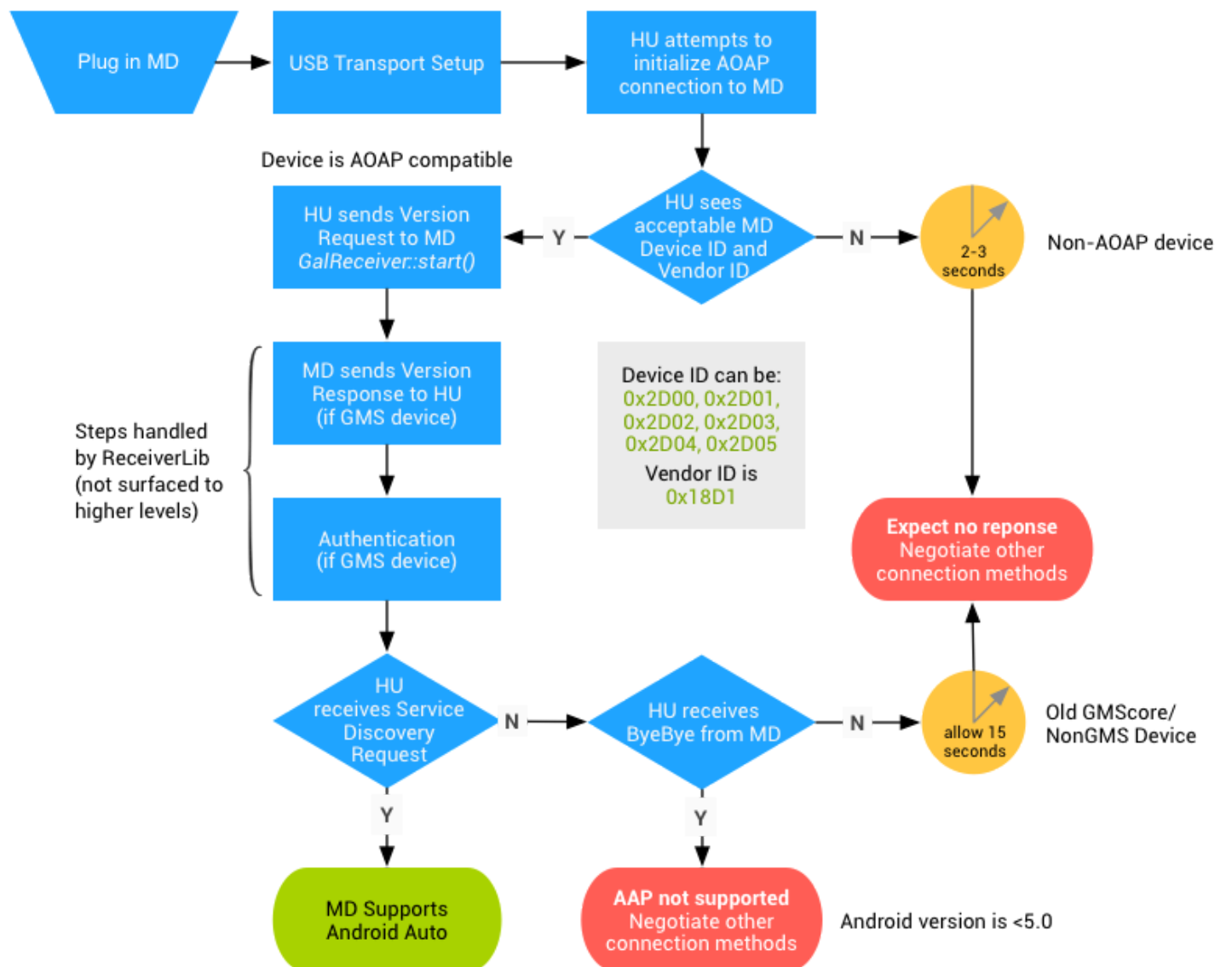


Figure 4.2. Overview of AAP connection flow.

## 4.4. Service Discovery Response

After registering the services it supports (Video, Audio, Bluetooth, etc.) and setting metadata, the HU calls the



Google Restricted `::start()` method to send the Service Discovery Response.

If an HU supports wireless projection, it MUST register a `WifiProjectionEndpoint` service (R04-075).

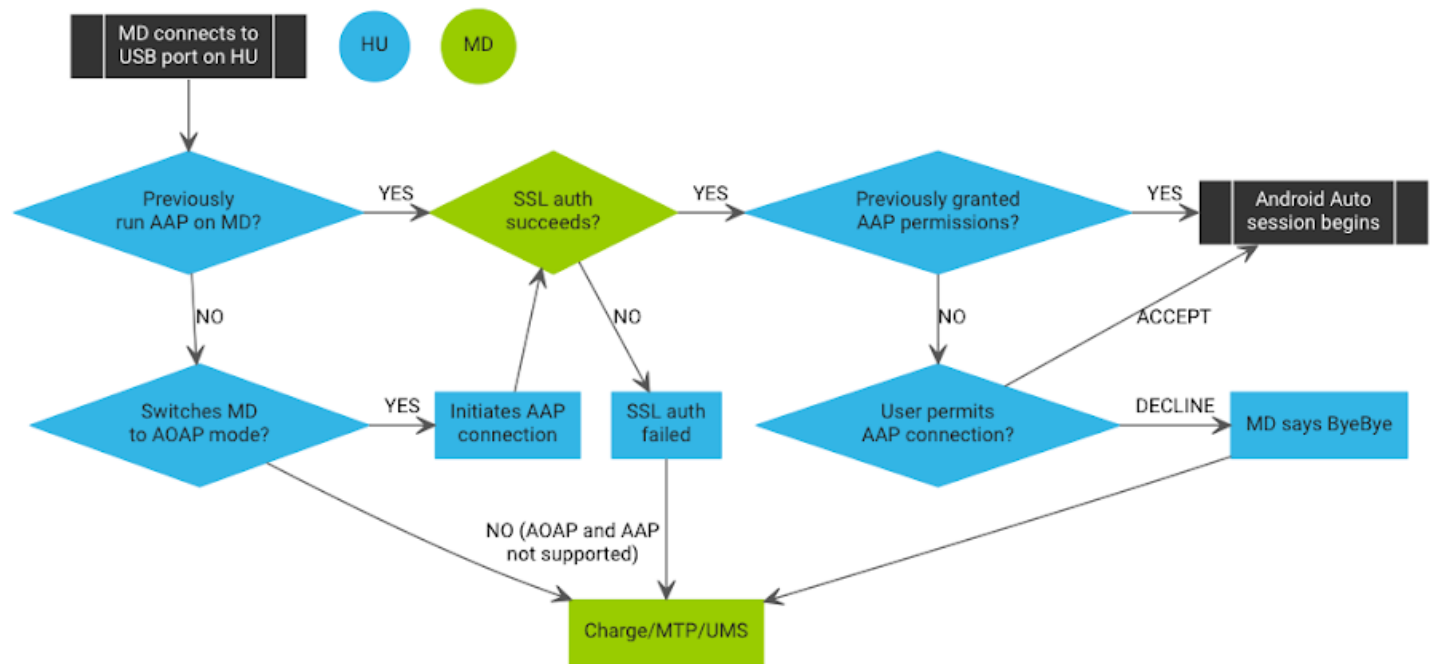


Figure 4.3. Authorizing AAP connection.

If an HU shows a dialog asking users if they want to connect AAP, the HU MUST NOT send the Service Discovery Response before the user grants permission to the HU to start AAP (R04-080).

Requirements for metadata included in the Service Discovery Response are described in the following sections. **MUST be provided** means NOT NULL and NOT EMPTY.

#### 4.4.1. Make

Vehicle marque. 'Car Company' rather than 'car\_company' or 'CarCompany' or other variants. Make:

- MUST be provided (R04-090)
- MUST match brand name of vehicle or HU for aftermarket (R04-100)
- MUST be consistent across Android Auto compatible products (R04-110)
- MUST be UTF-8 (R04-120)

#### 4.4.2. Model

Vehicle model (technical or code name suffices). Partner MUST provide Google with logic or look up table to parse model name if a code name is set (R04-130). Model:

- MUST be provided (R04-140)
- MUST be set to *After Market* if HU is an aftermarket HU (R04-150)
- MUST include the word *truck* at the end of the model string if the HU is shipping in a truck (R04-155)
- MUST be 7-bit ASCII (R04-160)

The term *truck* in this document is used to refer to a vehicle that requires a commercial driver's license to operate. Refer to [Vehicle Types](#) for more details on vehicle types and associated restrictions.

#### 4.4.3. Year

Model year, as distinct from year of manufacture (typically launch year). Year:

- MUST be provided (R04-170)
- MUST be a four (4) digit number if specific target year exists (R04-180)
- MUST be set to *multi* if specific target year does not exist (R04-190)

#### 4.4.4. Driver position



Indicates the position of the steering wheel, which may influence the position of UI elements. Choose one from the [Driver Position enum](#). Driver Position:

- MUST be provided (even if unknown) (R04-200)

#### 4.4.5. Head Unit Make

HU manufacturer (supplier) name or brand. Head Unit Make:

- MUST be provided (R04-210)
- MUST be 7-bit ASCII (R04-220)
- MUST be the same value as Make field if OEM does not utilize a separate supplier for this HU (R04-230)

#### 4.4.6. Head Unit Model

HU model, identifier, or platform code. Head Unit Model:

- MUST be provided (R04-240)
- MUST be 7-bit ASCII (R04-250)

#### 4.4.7. Head Unit Software Build

HU software build, typically an internal build identifier for HU software. Head Unit Software Build:

- MUST be provided (R04-260)
- MUST be 7-bit ASCII (R04-270)
- MUST be set to "multi" (all lower case letters) as the value if HU does not have specific build numbers (R04-280)

#### 4.4.8. Head Unit Software Version

HU software version. Typically a version identifier a vehicle owner can find in the Settings menu of the HU. Head Unit Software Version:

- MUST be provided (R04-290)
- MUST be 7-bit ASCII (R04-300)

#### 4.4.9. Vehicle ID

Identifier for the vehicle. An MD connecting to an HU can determine whether this is the first time the MD has been connected to this HU by comparing Vehicle IDs. Vehicle ID:

- MUST be unique string per HU and longer than 64 bits (R04-310)
- SHOULD be regenerated for every factory reset performed on the HU (R04-320)
- MUST NOT be a traceable ID with real-world significance (such as the VIN) (R04-330)

#### 4.4.10. Session configuration

The AAP UI includes the current time, signal indicators, and battery status. When AAP is displayed as part of a blended UI, the native components of the UI MUST NOT duplicate status information shown in the AAP UI (R04-340). The vehicle either doesn't display this status information or suppresses the AAP status icons using the Session Configuration settings.

#### 4.4.11. Display Name

Name populated in the AAP UI to indicate the entry point for users to return to native. Display Name:

- MAY be provided R04-350)
- MUST be consistent across Android Auto compatible products (R04-360)
- MUST be UTF-8 (R04-370)

If Display Name is not set, the value of the Make field will be used instead.

## 4.5. First run experience

On first-run (the initial AAP connection between an MD and a specific HU), the HU MUST NOT display more than one pop-up from the start trigger until the connection is established (R04-375). This applies to wired and wireless projection. Once a connection is established, the MD does not send video data until the user grants permission on the MD. During the first run, the HU MUST display a message to attract the user's attention to the MD screen (R04-380). A suggested implementation is:

- After establishment of an encrypted AAP connection, start a timer for 3 seconds.
- If the timer completes and `VideoSink::setupCallback()` has not yet been called (i.e. video feed not established), display a message instructing user to check the device.
- When the MD starts to send video frames, the projected display MUST be shown without further user interaction.

Suggested text for the message is: "To use Android Auto, please stop your car and follow the instructions on your phone". Variations on the phrasing of the message are allowed, but the spirit of the message MUST be maintained (R04-390). The prompt MAY be presented as a full-screen dialog box or as a dialog box that partially covers the current screen (R04-391). When transitioning to a prompt, the HU MUST NOT show an empty or blank screen and MUST NOT use distracting transitions or visual effects (R04-392).

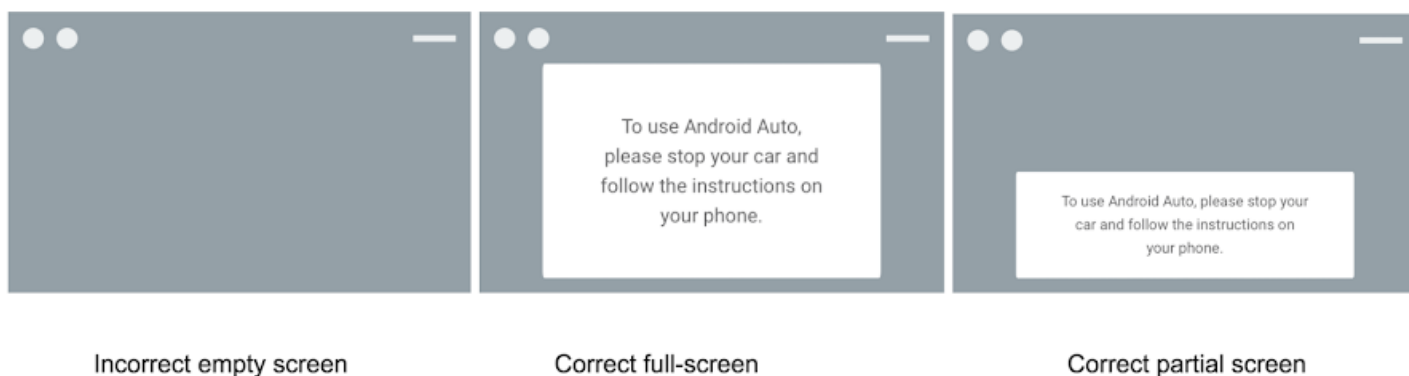


Figure 4.4. First run prompts.

On first-run, the HU MAY display safety and/or permission prompts (R04-400). On subsequent connections to the same MD, the HU MUST NOT display any setup steps or non-critical safety messages (R04-410).

## 4.6. Manage applications

The MD prompts the user to install, update, and configure required applications. This step occurs entirely on the MD and does not involve the HU. If all required applications meet minimum version requirements and are properly configured, this step is skipped.

## 4.7. Initial video focus

The MD enables screen projection to the HU, enabling the HU screen to show projection contents when the MD requests video focus.

On first-run, the MD requests video focus and the HU MUST grant it (as the end user is explicitly setting up Android Auto at this time) (R04-420). On subsequent connections, the MD will not request video focus. The HU SHOULD give the MD video focus upon connection without requiring the user to initiate any action (R04-430).

## 4.8. Run tutorial

On first-run the MD projects a tutorial that includes visible content and user interaction on the MD screen and on the HU screen.

## 4.9. Start Android Auto UI

At this point, the AAP connection process is complete, although Bluetooth pairing and HEP connection may still be

## 4.10. Disconnecting AAP

The HU MUST NOT disconnect AAP when the MD is connected to the HU, unless the user explicitly disconnects via a UI affordance or the MD terminates the connection (R04-440). When the HU switches between native and projection mode, HU MUST keep the AAP connection active even if the MD does not have video or audio focus (R04-450).

## 4.11. Handling concurrent connections

As the MD cannot know the state of other devices in the system, the HU MUST determine how to handle connections from multiple AAP compatible devices (R04-460). For example, during an active AAP connection between an MD and an HU, the HU implementation could handle a second AAP-compatible device being connected by leaving the first device connection intact and simply charging the second device.

## 4.12. Authentication certificates

The AAP protocol uses industry standard [TLS 1.2](#) with Client Authentication to secure communications between the HU (TLS Client) and MD (TLS Server). The first step in establishing an AAP connection is mutual authentication between the MD and HU following the TLS Client-authenticated TLS handshake protocol.

The AAP sender (on the MD) and the AAP Receiver Library (on the HU) perform mutual authentication using two (2) certificates signed by the Google Certificate Authority (CA):

- The HU certificate is sent to MD, which verifies it to ensure the HU is a valid AAP receiver.
- The MD certificate is sent to HU, which verifies it to ensure the MD is a valid AAP device.

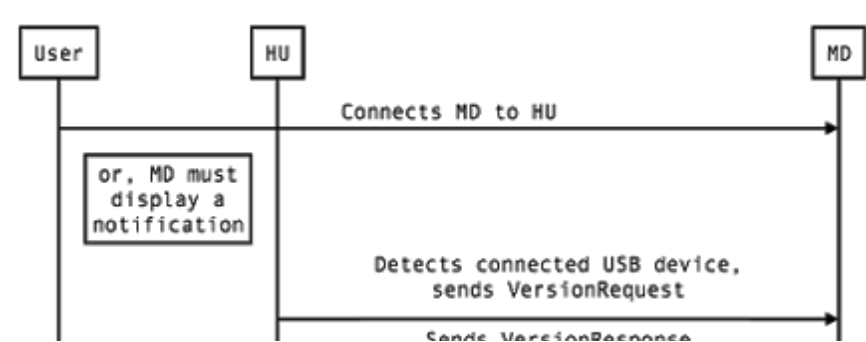
If authentication succeeds, the sender and Receiver Library initiate the AAP projection connection. If the Receiver Library cannot verify the sender certificate, the HU terminates the connection and disconnects AOAP (the MD might attempt to reconnect in another USB mode).

Google provides HU certificates to integrators after an HU has been certified.

Type	Format
Public/Private Key Format	RSA with 2048 bit keys
Digital Certificate Format	X.509 <a href="http://tools.ietf.org/html/rfc5280">http://tools.ietf.org/html/rfc5280</a>

## 4.13. Authentication sequence

The AAP Receiver Library uses the OS Abstraction layer API to retrieve the HU digital certificate and private key. After authentication completes, the Receiver Library notifies the HU of the MD identity and initiates the AAP protocol.



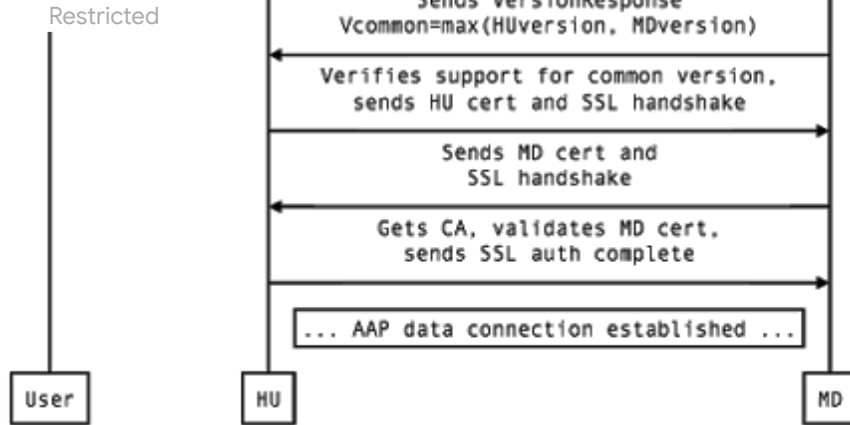


Figure 4.5. MD and HU establish trust using SSL certificates.

As a part of SSL authentication, the AAP Receiver Library compares the time and date information from the HU with the MD digital certificate validity period.

If the internal clock on the HU is incorrect and/or does not match the approximate time on the MD, SSL authentication might fail with error "Certificate not yet valid" or "Certificate expired". In case of SSL authentication failure, the ReceiverLib 'IControllerCallbacks::unrecoverableErrorCallback' is called with error code 'STATUS\_AUTHENTICATION\_FAILURE\_CERT\_NOT\_YET\_VALID'. The HU MUST inform the user about the reason for the SSL authentication failure and provide guidance on how to resolve the issue (R04-470). To resolve this error, the HU needs to send the correct time as obtained by GPS, Mobile Network/NITZ, NTP, set manually by user, etc.

## 5. Wireless connection

The AAP protocol v1.4 introduces wireless projection, i.e. AAP running over a Wireless Local Area Network (Wi-Fi) with the HU as the access point. Wi-Fi Direct is not supported.

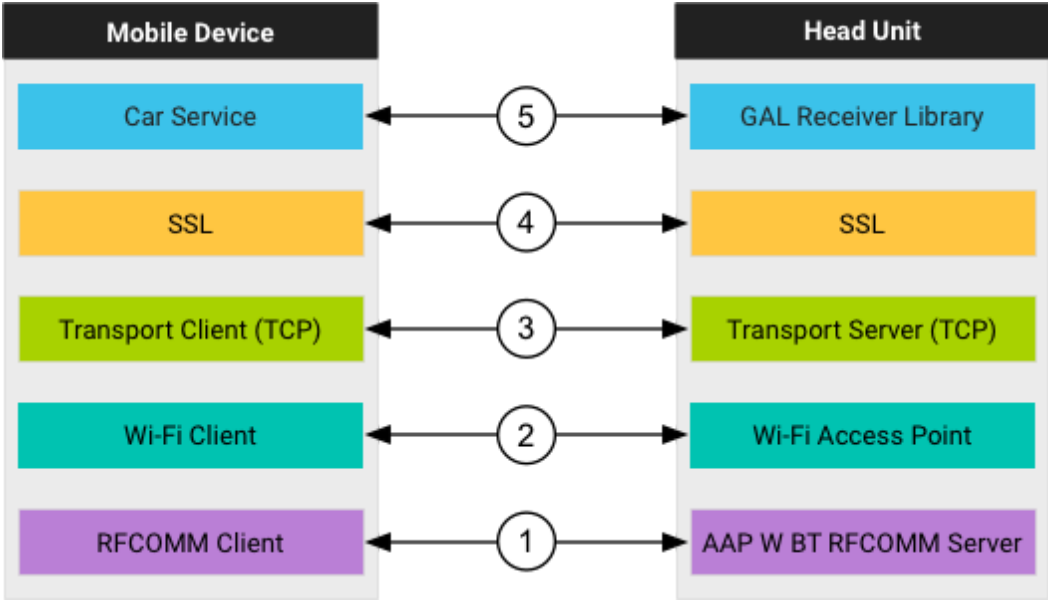


Figure 5.1. Sequence of steps in Android Auto Wireless.

### 5.1. Wireless hardware

If an HU supports AAP protocol v1.4 or later and meets the wireless hardware and latency requirements defined in this section, then it MUST support wireless projection and wired projection (R05-010). If an HU supports AAP Protocol v1.3 or earlier or does not meet the wireless hardware and latency requirements, then it MUST support only wired projection (R05-020).

HUs that support wireless projection are referred to as wireless HUs. The following requirements in this section are applicable only to wireless HUs. Wireless HUs MUST support 5 GHz 802.11ac (R05-025). Wireless 2.4 GHz 802.11b/g/n is currently NOT supported. Messages and fields to negotiate support for 2.4 GHz 802.11n have been added for support and compatibility in future.

Wireless HUs MUST provide a Wireless Local Area Network (Wi-Fi) with the HU being the access point (R05-030). The Wi-Fi network provided by the HU:

- MUST have a unique SSID for each HU (R05-031)
- MUST use Wi-Fi Protected Access II (WPA2) encryption (RSN/AES) to ensure the security of the connection to the MD (R05-032)
- MUST be protected by a unique Pre-Shared Key (PSK) generated randomly using a Cryptographically Secure Pseudo Random Number Generator (CSPRNG) upon the access point's first initialization (R05-033). The generated key MUST be at least 12 characters long and formed of the set of characters that are upper case, lower case and numbers (R05-034). The HU MUST provide a native user interface to allow the user to automatically generate a new PSK and/or manually change the current PSK (R05-035).
- MUST be advertised as 'ANDROID\_METERED' over the vendor-specific option 43 in the access point DHCP configuration if the access point has Internet access(R05-036). This is to enable the MD to suppress data exhaustive activities while AAW is active.
- MUST assign IPs dynamically to the connected devices by running a DHCP server on the Wi-Fi interface (R05-037)
- MUST assign only IPv4 addresses for wireless projection (R05-038). IPv6 is not currently supported.

### 5.1.1. Wireless hardware regulatory restrictions

The regulatory restrictions that devices (both HU and MD) must meet when utilizing Wi-Fi are outside the scope of this document. For current information and guidance from the Android Auto team on this topic, see [WLAN Connectivity](#).

### 5.1.2. Wireless latency requirements

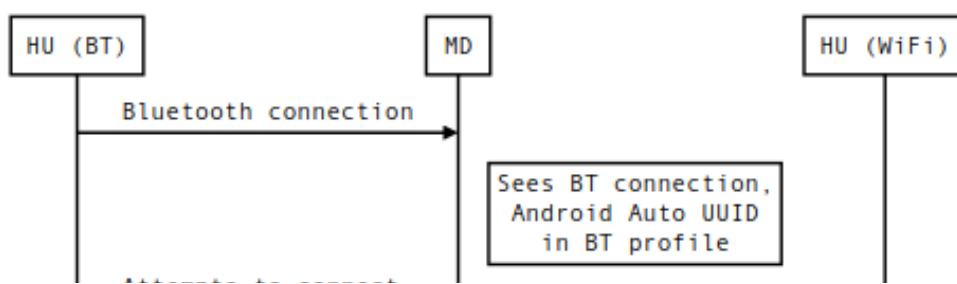
The connection over Wi-Fi between a wireless HU and MD MUST support a sustained throughput of at least 4 MBps while maintaining a Round Trip Time (RTT) of less than 200ms (R05-040). This wireless latency requirement is tested with a Bluetooth HFP phone call in progress and the wireless AAP connection heavily loaded. For detailed test conditions and instructions, refer to the [Wireless Latency Test](#) in the Android Auto Partner Help Center.

## 5.2. Wireless connection process overview

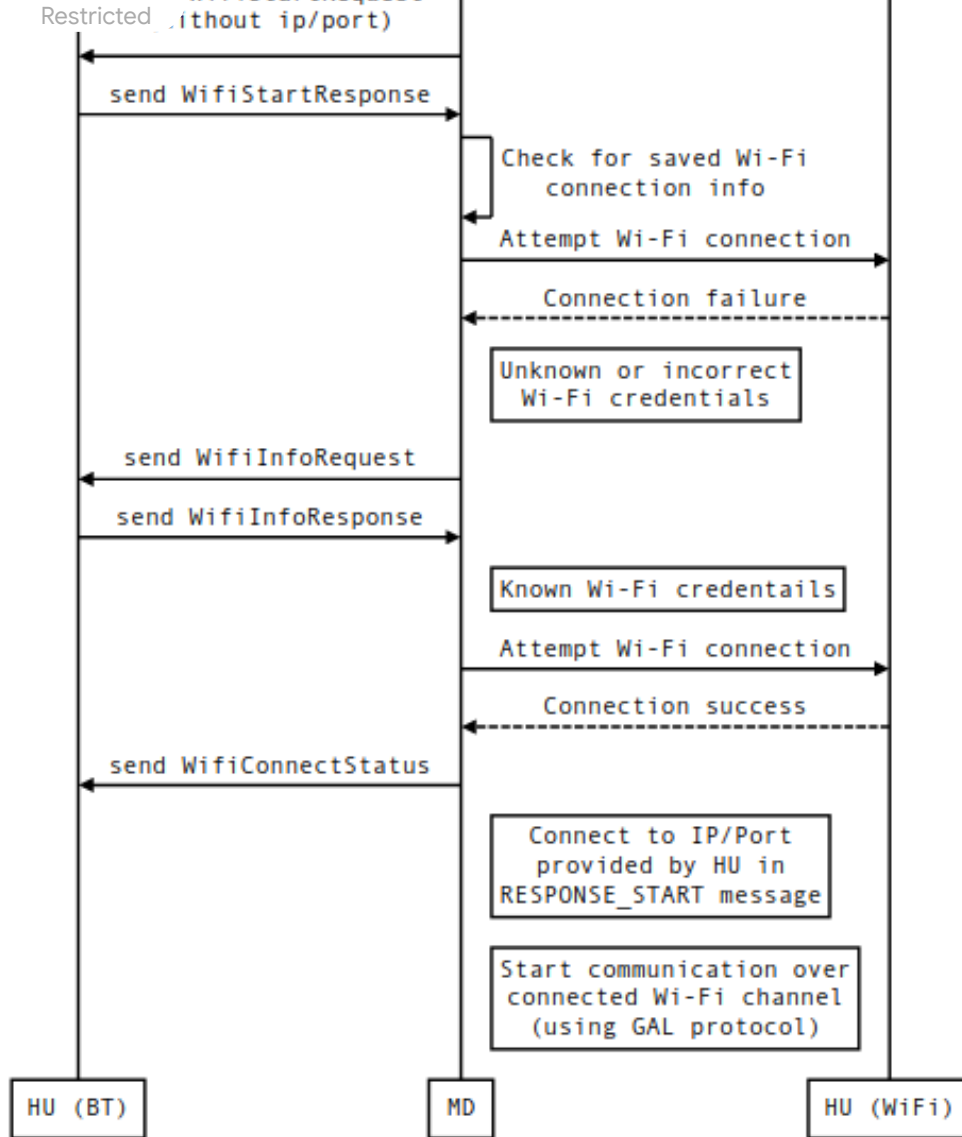
Wireless projection requires the establishment of multiple communication links between the MD and HU. At a high level, the wireless connection process includes the following steps:

1. Bluetooth service discovery
2. Bluetooth pairing
3. Bluetooth HFP connection
4. Bluetooth RFCOMM connection
5. Bluetooth RFCOMM version negotiation

At this point, wireless projection can be started from either the MD or HU. A wireless projection session is initiated by either the HU or MD sending a [WifiStartRequest](#). The connection process then continues with the following steps:







**Figure 5.3.** Wireless projection setup, Session start by MD

1. Wi-Fi credential exchange via Bluetooth RFCOMM (if required)
2. Wi-Fi network connection
3. TCP connection with the HU as server and MD as client
4. AAP service discovery process
5. Wi-Fi credential exchange (as part of AAP service discovery)
6. AAP session establishment (AAP over TCP/IP over Wi-Fi)

### 5.3. Bluetooth service discovery

To indicate that it supports wireless projection, a wireless HU MUST advertise the Android Auto UUID [4de17a00-52cb-11e6-bdf4-0800200c9a66](#) at all times it participates in Bluetooth service discovery (R05-050). A Wireless HU MUST NOT advertise more than 21 services in the service discovery phase, since this is the maximum number of UUIDs Android devices can parse at a time (R05-055).

### 5.4. Bluetooth RFCOMM control channel

A wireless HU MUST support version 1.0 of the Bluetooth Radio frequency communication (RFCOMM) protocol and implement a Bluetooth RFCOMM server (R05-060). The Android Auto [Partner Development Kit \(PDK\)](#) includes a sample RFCOMM implementation, and the [AAP Wi-Fi Discover Protocol Specification](#) documents the protocol. A wireless HU MUST support an RFCOMM connection over each active Bluetooth connection (R05-070).

An MD that supports wireless projection may initiate a RFCOMM connection to a wireless HU before or after a Bluetooth HFP connection is established and the wireless HU MUST be able to handle both scenarios (R05-075).



Restricted connection is established and the wireless HU MUST be able to handle both scenarios (R05-075). The HU MUST establish and maintain the RFCOMM connection even if a wired projection session is started and/or the Bluetooth connection itself is triggered by AAP as part of the AAP connection and launch process (R05-077).

Not all MDs that support wired projection support wireless projection. For current information on this topic, see [WLAN Connectivity](#).

#### 5.4.1. Bluetooth RFCOMM version negotiation

A wireless HU MUST send a **WifiVersionRequest** to the MD immediately after RFCOMM connection (within 1s) informing the MD the highest version of the wireless projection RFCOMM control channel it implements (R05-080). The **WifiVersionRequest** MUST include the frequency of the Wi-Fi channel that the HU's access point intends to broadcast on (R05-085). The **WifiVersionRequest** MAY include one 2.4 GHz frequency and one 5 GHz frequency if the access point broadcasts on both bands simultaneously (R05-086).

In response to a **WifiVersionRequest**, the MD will send a **WifiVersionResponse**. If the HU is broadcasting on a compatible frequency, the MD will send a **WifiVersionResponse** message with **version\_status** set to **SUCCESS**. If a compatible frequency is not found, the MD will send a **WifiVersionResponse** message with **version\_status** set to **STATUS\_NO\_SUPPORTED\_WIFI\_CHANNELS** to let the HU know that projection is not possible over the selected Wi-Fi channel. The HU MUST NOT send a **WifiStartRequest** until it has received a **WifiVersionResponse** message with **version\_status** set to **SUCCESS** (R05-088). After receiving a **WifiVersionResponse** message with **version\_status** set to **STATUS\_NO\_SUPPORTED\_WIFI\_CHANNELS**, the HU MAY send another **WifiVersionRequest** with a different Wi-Fi channel frequency to find a frequency supported by the MD (R05-089).

#### 5.4.2. Bluetooth RFCOMM WifiStartRequest

A wireless HU sends a **WifiStartRequest** to start wireless projection. A **WifiStartRequest** sent from HU to MD MUST contain the IP address and port number for the TCP connection that follows from the MD to the HU (R05-090).

Before sending a **WifiStartRequest**, an HU MUST enable its Wi-Fi Access Point and start its TCP server (R05-100). If an HU cannot enable its Wi-Fi Access Point without user interaction, it MUST NOT send a **WifiStartRequest** but instead display native guidance to the user on how to setup Wi-Fi (R05-110).

A wireless HU MAY implement a mechanism to automatically start wireless projection (by sending a **WifiStartRequest**) for devices that have been previously connected, etc. (R05-120). An MD may also send a **WifiStartRequest** to start wireless projection and the Wireless HU MUST attempt to start wireless projection as a result (R05-125). A **WifiStartRequest** sent from the MD to the HU does not contain an IP address or a port number.

##### 5.4.2.1. Bluetooth RFCOMM WifiStartRequest error handling

If the HU does not receive a **WifiStartResponse** within 25 seconds of sending a **WifiStartRequest**, the HU MUST display a native error message to the user informing them that the MD could not start wireless projection successfully (R05-130). A suggested error message is: "Unable to start Android Auto. Please try plugging in your phone via USB."

#### 5.4.3. Bluetooth RFCOMM WifiStartResponse

A **WifiStartResponse** message is sent from either the MD or HU in response to a **WifiStartRequest**. When an MD sends a **WifiStartResponse** message, it will contain only a status message.

A **WifiStartResponse** message sent from a wireless HU:

- MUST be sent immediately (within 1s) of receiving a **WifiStartRequest** (R05-140)
- MUST contain an IP address and port number for the TCP connection that follows from the MD to the HU (R05-150)
- MUST have status set to **STATUS\_PROJECTION\_STARTED\_ALREADY** if a projection session is already active between the HU and any device (R05-160)



- MUST have status set to **STATUS\_WIFI\_DISABLED** if Wi-Fi is disabled on the HU and cannot be enabled automatically (R05-165)
- MUST have status set to **STATUS\_WIFI\_NOT\_YET\_STARTED** if the HU is able to enable Wi-Fi but has not yet completed this task (R05-170)
- MUST have status set to **STATUS\_SUCCESS** if the HU is ready to proceed to next step of the connection process (R05-180)

If an HU sends a **WifiStartResponse** message with status set to **STATUS\_WIFI\_DISABLED**, it MUST also display native guidance to the user on how to setup Wi-Fi (R05-190).

If the MD receives a **WifiStartResponse** with status set to **STATUS\_WIFI\_NOT\_YET\_STARTED**, it will continue to send further **WifiStartRequest** messages with an exponential backoff until a **WifiStartResponse** message with a different status is received.

The Android Auto Bluetooth RFCOMM control channel specification does not contain a message to stop projection as this is handled via the **ByeBye** message defined in the AAP protocol.

#### 5.4.4. Bluetooth RFCOMM WifiInfoRequest and WifiInfoResponse

A wireless HU MUST support exchange of Wi-Fi credentials via both the AAP protocol as part of Service Discovery and via Bluetooth RFCOMM communication (R05-200). A **WifiInfoRequest** message may be sent from the MD to HU to request the HU to send Wi-Fi credentials over RFCOMM if the MD does not already have valid Wi-Fi credentials.

A **WifiInfoResponse** message sent from a wireless HU:

- MUST specify a **wifi\_ssid** as an ASCII string (R05-210)
- MUST specify a **wifi\_password** as an UTF-8 string (R05-220)
- MUST specify a **wifi\_bssid** in Ethernet MAC address format (R05-230). If the wireless HU has two access points with the same SSID on different frequency bands, the BSSID of the 5 GHz access point MUST be utilized (R05-235).
- MUST be sent immediately (within 1s) after receiving a **WifiInfoRequest** (R05-240)

#### 5.4.5. Bluetooth RFCOMM WifiConnectStatus

When an MD attempts to connect to an HU over Wi-Fi, it will notify the HU of Wi-Fi connection status by sending a **WifiConnectStatus** message. If the HU receives a **WifiConnectStatus** with status of **STATUS\_WIFI\_INACCESSIBLE\_CHANNEL** it MAY display a native error message giving instructions to the user on how to fix or troubleshoot the issue (R05-250). A suggested error message is: "Your Android phone is unable to connect to the wireless access point in your vehicle. Please check wireless settings."

#### 5.4.6. Bluetooth RFCOMM error handling

If a read error occurs in the HU Bluetooth RFCOMM socket, the HU MUST wait for the MD to reconnect RFCOMM and then send the MD a **WifiVersionRequest** (R05-260).

If the MD does not connect to the HU's access point within 60 seconds of receiving the **WifiStartRequest**, the HU MAY display a message asking the user to restart Bluetooth on the MD (R05-265).

#### 5.4.7. Ending Bluetooth RFCOMM connection

The Bluetooth RFCOMM connection MUST NOT be ended for the duration of the Bluetooth HFP connection between the HU and MD, even if a projection session has ended (R05-270).

#### 5.4.8. Naming devices in native screens

An MD will have the following names associated with it (in order of preference):

1. **Bluetooth device name**. Advertised by the MD as part of Bluetooth service discovery and is user editable, e.g.

2. **AAP device\_name**. Sent by the MD in its **ServiceDiscoveryRequest** during AAP service discovery and is not user editable, e.g. *Google Pixel 2*.
3. **USB device manufacturer and product**. Available after a device enumerates itself via USB. Not user editable, e.g. *Google Pixel 2*.

The HU MUST use the most preferred name available when showing a device in a native screen (R05-530).

## 5.5. TCP connection

A wireless HU MUST implement a TCP server (R05-280). An HU MUST be able to accept incoming TCP connection from a second device while a projection (wireless or wired) is active with a first device to enable seamless wireless projection user switching (R05-290). An HU MUST be able to end a projection session without shutting down the TCP server (R05-300).

The TCP server MUST set the following connection socket settings (R05-310):

- Read timeout = 5 seconds
- recv buffer size = 16K
- Nagle options = tcp no delay
- TCP sockets in established states = graceful close

When the wireless projection session is initiated from the HU, the TCP connection MUST be closed only for read then wait for existing send data to be sent before closing the socket (R05-312). After a TCP connection is established, the HU initiates negotiation of the AAP connection by sending a Version Request to the MD with the ReceiverLib function **GalReceiver::start()**. For details on the AAP connection process, refer to [Connect AAP](#).

### 5.5.1. TCP connection error handling

If the HU does not receive a TCP connection request from the MD within 45 seconds from sending the **WifiStartRequest**, the HU MAY display a popup to prompt the user to restart Bluetooth and Wi-Fi on the MD (R05-315).

If the HU has not sent or received a **ByeByeRequest** to/from the MD and detects a TCP socket read timeout or receives a reset (RST) packet, the TCP server MUST close its open socket and the HU MUST display a native message stating that the MD has disconnected (R05-320). A suggested error message is: "A wireless connection error has occurred with your phone. Please reconnect or try plugging in via USB."

### 5.5.2. Ending TCP connections

If the HU receives a **ByeByeRequest** from the MD, the TCP server on the HU MUST close its open socket only after it has sent a **ByeByeResponse** message (R05-330).

## 5.6. Starting wireless projection

For details on how the HU must handle a user actuating an Android Auto Launch Icon shown on native screens, refer to [Launch Icon actuation](#).

After sending a **WifiStartRequest** to an MD, a wireless HU MUST NOT send a second **WifiStartRequest** request to a different MD before it receives a **WifiStartResponse** from the first MD (R05-340).

If a wireless HU receives multiple concurrent **WifiStartRequest** messages from different MDs, it MUST send a **WifiStartResponse** with status set to **STATUS\_PROJECTION\_STARTED\_ALREADY** to the MD(s) that it does not intend to start projection with (R05-350).

A wireless HU MUST NOT attempt to start both wired and wireless projection with the same device and MUST NOT attempt to switch an MD to AOA mode while it is projecting over wireless (R05-360). The **WifiVersionResponse** message sent from the MD include the USB serial number of the MD to enable the HU to perform this check.

Specifically, an HU:

- MUST NOT send a **WifiStartRequest** when wired projection is already active or in the process of setting up with the same MD (R05-370).
- MUST NOT attempt to switch MD USB connection to accessory mode when wireless projection is already active or in the process of setting up (R05-375)
- MUST automatically start wireless projection on subsequent connections, if the user sets up AAP over Wi-Fi (R05-383)
- MUST not automatically start wireless projection on subsequent connections, if the user sets up AAP over USB (R05-385)

The MD saves Wi-Fi credentials and uses them for future connections to the same HU (based on the HU Bluetooth MAC Address). Error cases where saved credentials are no longer valid are handled by the MD, and the MD will issue a **WifiInfoRequest** request over Bluetooth RFCOMM or a **WiFiCredentialsRequest** over AAP to receive updated credentials when required.

## 5.7. Wireless projection error handling

The nature of wireless projection requires a wireless HU to handle a range of errors.

### 5.7.1. Wireless projection disconnect

A wireless HU MUST send a **PingRequest** message to the MD every second (R05-380) and MUST measure and track *ping round-trip-time (Ping RTT)* as the elapsed time between sending a **PingRequest** message and receiving a **PingResponse** (R05-390). The Logic to track Ping RTT is not part of the receiver library and MUST be implemented on the HU side (R05-395).

If Ping RTT exceeds three (3) seconds when a wireless projection session is active (including the case when a **PingResponse** is never received), the HU MUST display a native message stating that the MD has disconnected (R05-400). A suggested error message is: "The wireless connection with your phone has disconnected. Please reconnect or connect via USB."

If the HU receives a **ByeByeRequest** with the reason being **USER\_SELECTION** or **DEVICE\_SWITCH**, the HU MUST return to a native screen and be able to restart wireless projection without disconnecting the Wi-Fi connection or the Bluetooth connection with the MD (R05-405).

A wireless HU MUST NOT end the active wireless projection session even if the Bluetooth connection with the projecting MD is disabled (R05-407).

### 5.7.2. Wireless projection network connectivity issues

If an HU has video focus set to native and the Ping RTT is above 200ms for five (5) consecutive **PingRequest** messages, the HU MUST display a native message stating that Wi-Fi network connectivity is poor and asking the user to improve placement of the MD (R05-410). A suggested error message is: "The wireless connection with your phone is unstable. Please move your phone closer to the access point or connect via USB."

### 5.7.3. AAP BatteryStatusNotification

The AAP protocol v1.4 introduces a **BatteryStatusNotification** message sent from the MD to the HU. If an HU has video focus set to native and receives a **BatteryStatusNotification** with **critical\_battery** set to TRUE, the HU MUST display a native message stating that the MD has low battery and should be plugged in or charged immediately (R05-420). This native message SHOULD include the current battery level and estimated time remaining in seconds (R05-430). A suggested error message is: "Your phone is running low on battery. Please plug it in or charge it as soon as it is safe to do so."

If an HU has video focus set to projected and receives a **BatteryStatusNotification** with **critical\_battery** set to TRUE, the HU MUST NOT display a native message as the Android Auto UI will display a message (R05-440).

In case an HU implements a battery-low notification that relies on the battery indicator parameter in the +CIEV result code of the BT HFP profile, the HU MUST suppress this notification for the MD running AAW if the MD has video focus (R05-445).

## 5.8. Switching projection sources

The user switching APIs previously documented in HUIG v2.3.1 should be considered experimental and not included in production HU (R05-575). If multiple MDs are available for projection (including any devices actively projecting), the HU:

- MUST NOT support multiple concurrent projection sessions (R05-450)
- MUST provide a native screen that enables a user to switch between projection sources (R05-510)
- MUST have the device switching UI accessible in no more than two (2) taps from the AA active session (R05-511)
- MUST retain the link between the Android Auto icon and the current projection session (R05-512)
- SHOULD provide a device switcher widget on the native home screen (R05-513)

## 6. Bluetooth

To provide a consistent hands-free telephony experience across both the native and projection user interfaces, AAP uses Bluetooth Hands-Free Profile (HFP) for voice telephony communication.

The HU:

- MUST support Bluetooth HFP v1.5 or later (R06-010)
- MUST allow the user to use AAP even when Bluetooth HFP is not connected (R06-020)
- MUST allow the user to use Bluetooth on other mobile devices even when AAP is active (R06-023).
- MUST NOT require user interaction during the Bluetooth pairing and connection process (except in the case where the upper limit to the number of supported paired Bluetooth devices has been reached) (R06-021)

An HU that supports wireless projection MUST support multiple simultaneous Bluetooth RFCOMM connections to enable seamless wireless projection user switching (R06-025).

### 6.1. Bluetooth service announcement

As part of its Service Discovery Response, the HU MUST include a BluetoothService that specifies the HU Bluetooth MAC address and the HU-supported Bluetooth pairing methods (R06-030).

### 6.2. Protocol messages for Bluetooth pairing

After receiving the Service Discovery Response, the MD and HD use the [BluetoothPairingRequest](#), [BluetoothPairingResponse](#), [BluetoothAuthenticationData](#), and [BluetoothAuthenticationResult](#) messages to perform Bluetooth pairing.

#### 6.2.1. BluetoothPairingRequest

The [BluetoothPairingRequest](#) is sent by the MD to request exchange of Bluetooth pairing information. The MD may send this message at anytime to fix a corrupted state. This message includes the MD Bluetooth MAC address and the pairing method that will be used by the device.

If Bluetooth is disabled on the HU when it receives a [BluetoothPairingRequest](#), the HU MUST enable Bluetooth (R06-040).

#### 6.2.2. BluetoothPairingResponse

The `BluetoothPairingResponse` is sent by an HU in response to a `BluetoothPairingRequest`. The HU:

- MUST send a `BluetoothPairingResponse` within one (1) second of receiving a request (R06-050)
- MUST be able to handle multiple `BluetoothPairingRequest` messages in the same AAP connection (as requests may be re-sent anytime by an MD attempting to fix a corrupted state) (R06-060)
- MUST send a `BluetoothPairingResponse` with `STATUS_SUCCESS` as the status if it is ready to pair, or will be ready to pair within one (1) second of receiving the `BluetoothPairingRequest` (R06-070)
- MUST send a `BluetoothPairingResponse` with `STATUS_BLUETOOTH_PAIRING_DELAYED` as the status if cannot be ready to pair within one (1) second of receiving the `BluetoothPairingRequest` (R06-080)
- MUST send a `BluetoothPairingResponse` with `STATUS_SUCCESS` as the status if a `BluetoothPairingResponse` with `STATUS_BLUETOOTH_PAIRING_DELAYED` was sent previously and the HU is now ready to pair (R06-085)
- MUST send a `BluetoothPairingResponse` even when already paired with the MD (R06-090)
- MUST set the `already_paired` field in the `BluetoothPairingResponse` to true if the HU already has a link key (pairing information) for the MD that sent the request (R06-100)

After receiving a `BluetoothPairingResponse` with `STATUS_SUCCESS` as the status code, the MD initiates the pairing and HFP connection process with the HU. If each side has the link key (pairing information) for the other side, the pairing step is skipped and the MD initiates the HFP connection only.

When the MD receives a `BluetoothPairingResponse` with `STATUS_BLUETOOTH_PAIRING_DELAYED` as the status code, it ignores the `already_paired` field and waits for the HU to send another `BluetoothPairingResponse` with `STATUS_SUCCESS` as the status before initiating pairing and connection.

### 6.2.3. BluetoothAuthenticationData

When the MD initiates pairing, the HU MUST send the `BluetoothAuthenticationData` message to the MD to provide it with the required authentication data for Bluetooth pairing (R06-110). Authentication data consists of a UTF-8 character encoded string containing the passkey or numeric comparison value and the Bluetooth pairing method used by the HU Bluetooth stack.

### 6.2.4. BluetoothAuthenticationResult

The `BluetoothAuthenticationResult` is sent by the MD to indicate whether it was able to authenticate with the provided authentication data in the `BluetoothAuthenticationData` message. This message includes the result of the Bluetooth pairing process. The HU:

- MUST confirm the Bluetooth pairing when a `BluetoothAuthenticationResult` message is received with `STATUS_SUCCESS` (R06-115)
- MUST NOT confirm the Bluetooth pairing until a `BluetoothAuthenticationResult` message is received with `STATUS_SUCCESS` (R06-116)
- MUST drop the Bluetooth pairing and show an error message indicating pairing failure if a `BluetoothAuthenticationResult` message is received with any status other than `STATUS_SUCCESS` or if 30 seconds have elapsed since the `BluetoothAuthenticationData` message was sent to the MD (R06-117).

The HU MUST keep the AAP session active even if the Bluetooth pairing fails (R06-118).

## 6.3. Bluetooth pairing scenarios

The HU must consider several scenarios for Bluetooth pairing.

### 6.3.1. Pairing during a call

If the HU receives a `BluetoothPairingRequest` when a separate already connected MD is engaged in a Bluetooth HFP call, the HU MUST send a `BluetoothPairingResponse` with status set to `STATUS_BLUETOOTH_PAIRING_DELAYED` if it is unable to pair with a new MD during an active call (R06-120)



Restricted  
6.3.2.2. If it is unable to pair with a new MD during an active call (R06-120). After the call ends, the HU MUST send a **BluetoothPairingResponse** with status set to **STATUS\_SUCCESS** to the MD (to indicate the HU is ready perform Bluetooth pairing) (R06-130).

### 6.3.2. Pairing when limit of Bluetooth-paired devices is reached

Many HUs have an upper limit to the number of supported paired Bluetooth devices. The HU MUST NOT block pairing when the paired device limit has been reached (R06-140). When the upper limit of paired devices is reached, the HU MUST either automatically drop or guide the user to drop a pairing (R06-150). The HU SHOULD automatically drop the least recently used pairing (R06-160). If the HU does not automatically drop the least recently used pairing, a message must be displayed to users informing them to manually remove a pairing.

### 6.3.3. Pairing during an active AAW session

A wireless HU always maintains an active Bluetooth RFCOMM connection to a wirelessly projecting MD. The HU MUST be able to pair with new phones while maintaining the Bluetooth RFCOMM connection to the projecting MD (R06-165).

## 6.4. HFP connection

After pairing, the MD attempts to connect to Bluetooth HFP. The HU must allow the HFP connection from the MD (R06-170). If the HFP connection process is already in progress before a user plugs in a phone to start AAP, the HU MUST continue the HFP connection process (R06-180).

When the MD is already on a non Bluetooth call, the Bluetooth pairing and HFP connection process proceeds as normal. If the MD is already connected via HFP to a different HFP device than the HU and on a call when the AAP connection process is initiated, the MD will disconnect from the other HFP device and initiate pairing and connection to the HU.

If the MD is already connected via HFP to the HU and on a call when the AAP connection process is initiated, no further pairing or HFP connection is required, but the HU MUST allow the AAP connection to proceed without waiting for the call to end (R06-190).

If an HU can support only one HFP connection, it MUST NOT connect any other HFP devices while AAP is active except for user switching scenarios(R06-200).

## 6.5. Bluetooth audio (A2DP) connection

If the MD is connected to an Advanced Audio Distribution Profile (A2DP) device (including the HU) when connecting AAP, the MD leaves the A2DP connection intact. However, the HU MUST disconnect A2DP if an AAP session is started on a device connected via A2DP (R06-210). A2DP connections with other devices that are not connected via AAP MAY be left active (R06-220).

## 6.6. Bluetooth reconnection

If Bluetooth disconnects or is lost during an active AAP connection, the MD continuously attempts to connect to HFP. If Bluetooth disconnects, the HU MAY attempt to connect HFP to the AAP device, but MUST NOT connect to another Bluetooth device (R06-230). If Bluetooth pairing is lost or reconnect attempts repeatedly fail, the MD MAY send a **BluetoothPairingRequest** and then re-initiate Bluetooth pairing.

## 6.7. Phone calls from other HFP devices or embedded SIMs

An HU MAY allow a phone call to be made from a second connected HFP device or an embedded SIM when an AAP session is active (R06-240). When starting a call from a second connected HFP device or an embedded SIM, the HU MUST call the Receiver Library **CallAvailabilityStatus** message with **call\_available** set to **false** (R06-250). When ending a call from a second connected HFP device or an embedded SIM, the HU MUST call the Receiver Library **CallAvailabilityStatus** message with **call\_available** set to **true** (R06-260). When starting an AAP session while on a call with a second connected HFP device or an embedded SIM, the HU MUST send the MD a **CallAvailabilityStatus** message with **call\_available** set to **false** immediately after completing AAP session (R06-270).

## 6.8. Clarifications

While AAP needs a Bluetooth HFP connection for telephony to be routed through the HU during an active AAP session, the absence of an active HFP connection will not prevent incoming or outgoing phone calls. Not every MD will have an active Bluetooth connection. For example, the user may have explicitly turned off Bluetooth during an AAP session. AAP uses Bluetooth HFP for voice telephony communication but does not use Bluetooth for speech recognition (via Bluetooth voice recognition activation, or BVRA) or media playback audio streaming (via A2DP).

## 6.9. Additional Bluetooth profiles

AAP uses only Bluetooth HFP. During an AAP session, the HU may maintain other additional Bluetooth profiles (except A2DP). For example:

- Message Access Protocol (MAP)
- Phonebook Access Protocol (PBAP)
- Personal Area Network (PAN)
- Remote SIM Access Protocol (RSAP)
- Radio Frequency Communication (RFCOMM)

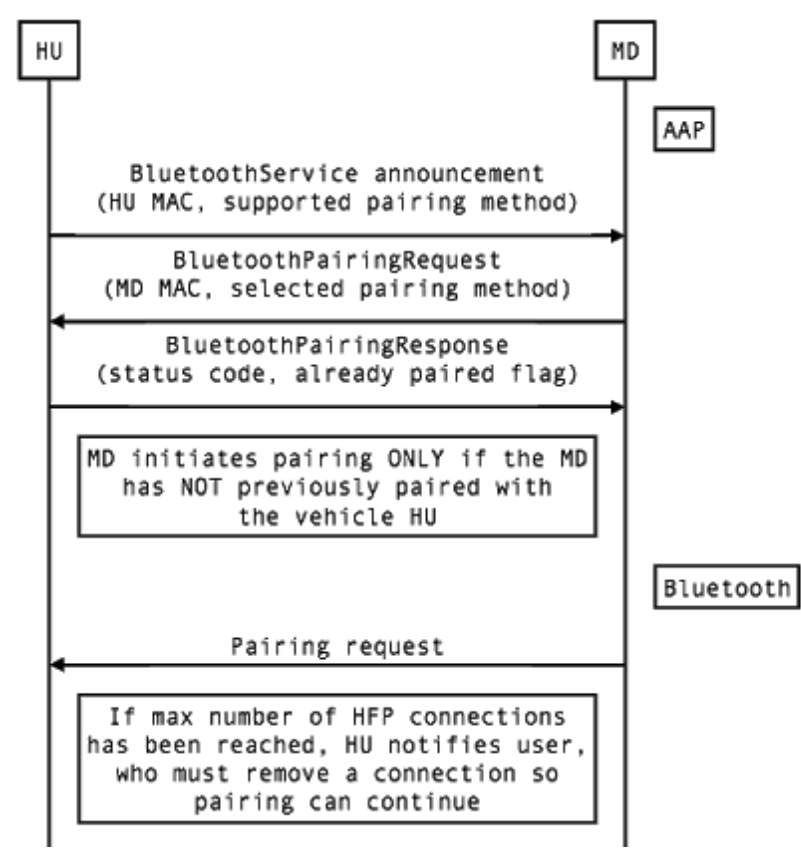
## 6.10. Bluetooth service announcement

If the HU wants the MD to skip the Bluetooth pairing and connection process, the HU can declare its Bluetooth MAC address as SKIP\_THIS\_BLUETOOTH (a useful trick when development and debugging). AAP supports both PIN and numeric comparison as pairing methods.

## 6.11. Bluetooth pairing and connection

After establishing AAP connection, the MD can automatically pair with the HU over Bluetooth.

If the MD has **never** paired with the HU before, the HU and MD use the following Bluetooth pairing and connection flow:



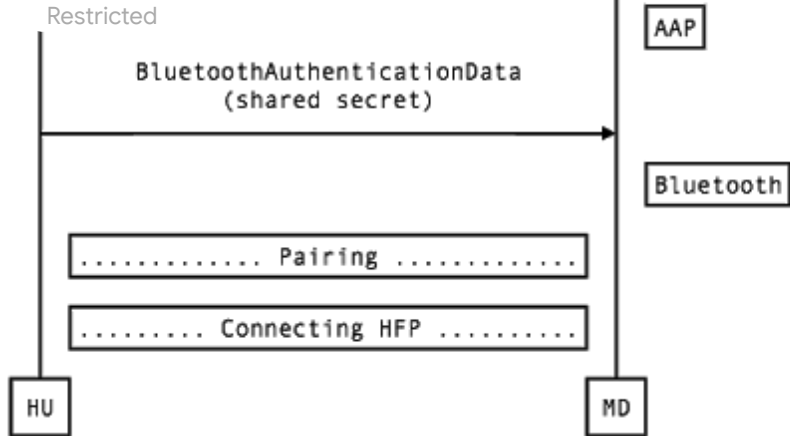


Figure 6.1. Bluetooth pairing authentication and connection flow when MD has never paired with vehicle.

If the MD has **previously paired** with the HU, the HU and MD use the following Bluetooth pairing and connection flow:

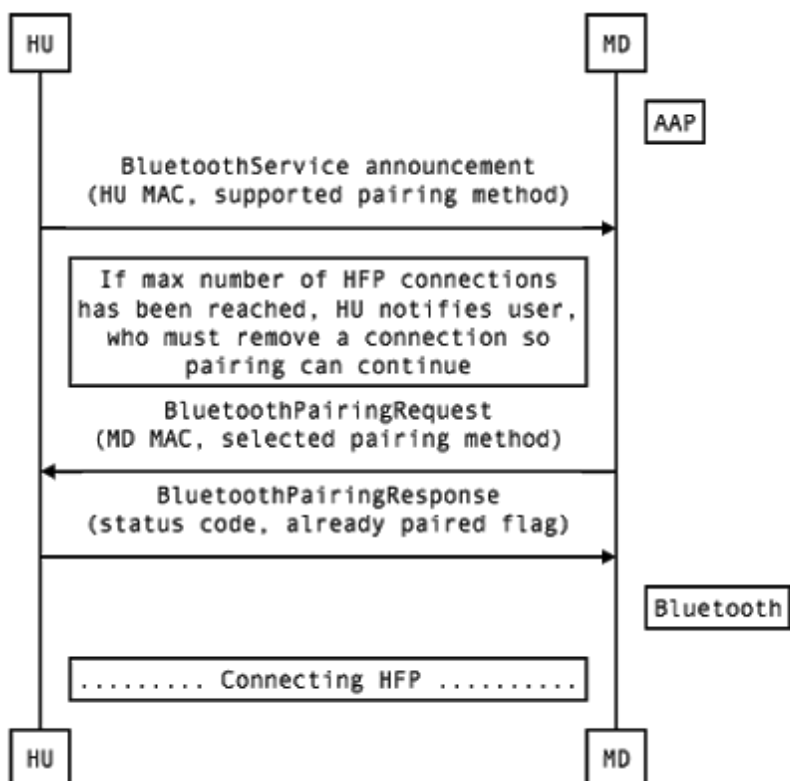


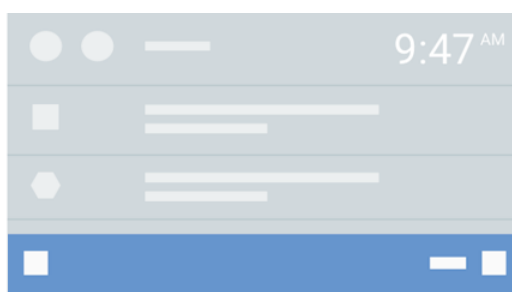
Figure 6.2. Bluetooth connection flow when MD has previously paired with vehicle.

## 7. Video

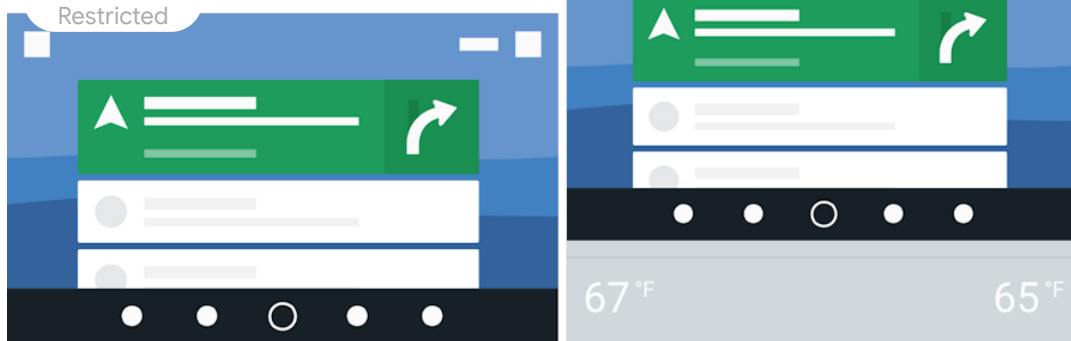
This section details requirements for AAP video integration.

### 7.1. Integration types

In projection mode, the HU displays a video stream encoded by the MD. The HU **MUST** display AAP content either full-screen or as part of a blended UI (R07-001).

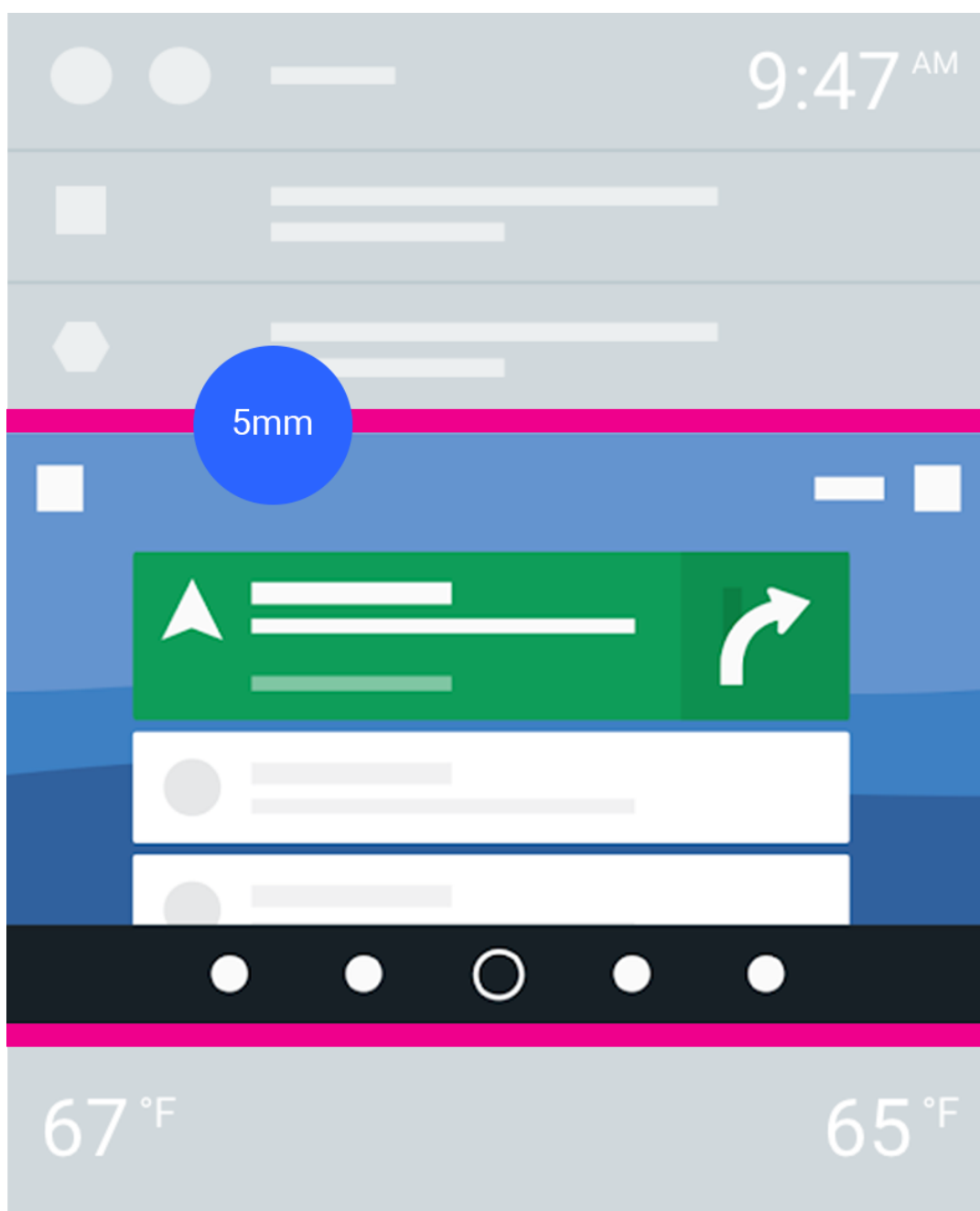






**Figure 7.1.** Examples of full-screen and blended UIs.

A blended UI displays AAP in a section of the screen simultaneously with native components. The HU **SHOULD** display AAP full-screen with projection covering the entire screen (R07-002). If vehicle controls for critical features such as advanced driver-assistance systems (ADAS), heating, ventilation, and air conditioning (HVAC), and other safety features are accessible only via on-screen controls, the HU **SHOULD** use a blended UI (R07-003). In a blended UI, if the native UI has touch, there **MUST** be at least 5mm of space between any native content and AAP content (R07-004).



**Figure 7.2.** Blended UI 5mm clearance.

Android Auto may display status information such as time, signal indicators, battery or other on-screen status information. The HU **MUST NOT** duplicate on-screen status information. The HU **MUST** suppress its own status bar component or suppress AAP status icons so information is not duplicated (R07-006).

## 7.2. Video decoder

The HU MUST provide an H.264/AVC Baseline Profile hardware decoder (R07-010). The HU hardware decoder:

- MUST support H.264 BP level 3.1 (R07-020)
- MUST support H.264 BP level 3.2 if the HU supports 720p (R07-030)
- MUST support H.264 BP level 4.2 if the HU supports 1080p (R07-040)
- MUST support a minimum frame rate of 30 FPS for all supported video resolutions (R07-050)
- SHOULD support a maximum frame rate of 60 FPS for all supported video resolutions (R07-060)
- MUST support all frame rates (continuous and noncontinuous) between 5 FPS and the max frame rate (R07-070)
- MAY support [Flexible Macroblock Ordering](#) (FMO) (R07-080)
- MAY support [Arbitrary Slice Ordering](#) (ASO) (R07-090)
- MAY support Redundant Slices (RS) (R07-100)

## 7.3. Frame rate

We strongly recommend that HUs use a hardware video decoder capable of rendering video of at least 60 frames per second (FPS) to match the typical MD 60 FPS UI redraw rate. For lower-end hardware, HUs can specify a 30 FPS video stream frame rate. The higher the frame rate, the lower the latency for input events (especially for touch gestures). Reducing latency will result in a better end-user experience.

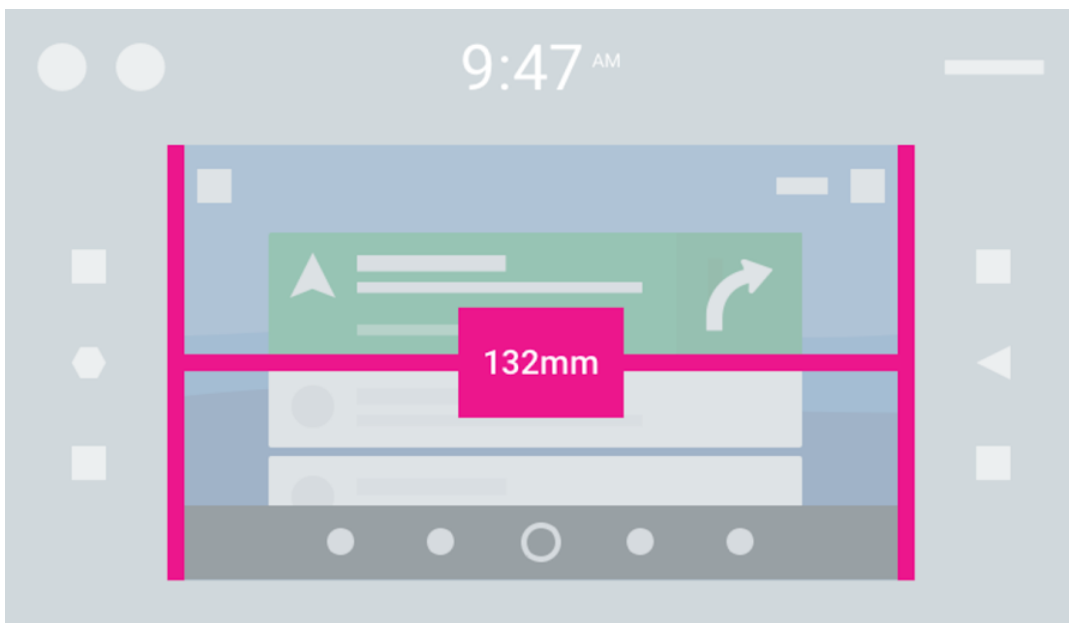
At runtime, the MD selects from supported resolutions and frame rates to determine actual frame rate. Each supported resolution MUST be reported with the maximum applicable frame rate (R07-500). Selection depends on many factors such as device power state and the order of advertised video configurations. To ensure high resolutions are utilized, the HU MUST advertise the supported video configurations with the highest resolution first (R07-510).

The MD may send frames at any rate lower than or equal to the specified frame rate to optimize system performance, e.g., from 30 FPS to 29 FPS, or directly from 15 FPS to 5 FPS.

## 7.4. Display area and screen

If an HU has a touchscreen, the AAP display area:

- MUST be at least 80mm high excluding any screen area covered by a bezel (R07-110)
- MUST be at least 132mm wide excluding any screen area covered by a bezel (R07-120)



**Figure 7.3.** UI projection area dimensions.

Minimum display area is based on minimum text size and tap target size requirements. Specifically, the minimum text size for secondary text on the Android Auto UI **MUST** be at least 12 arcmin (R07-121). (A capital H in the second line of text in a list is rendered at the secondary text size.) 64dp tap target size on touchscreens **MUST** be greater than or equal to 11.3 mm square (R07-122).

If an HU has a rotary controller and does not have a touchscreen, the minimum size is dependent on viewing distance and partners **MUST** use the [Screen Size Calculator](#) to definitively determine compatibility (R07-130). Input parameters are (a) diagonal screen size, (b) resolution, (c) pixel aspect ratio, and (d) viewing distance.

The HU screen:

- **MUST** support 16-bit color (R07-140)
- **SHOULD** support 24-bit color (R07-150)
- **MUST** have a pixel aspect ratio between 0.9 and 1.15 (R07-160)
- **MUST** have a large enough horizontal and vertical resolutions to generate a valid value through the [Screen Size Calculator](#) (R07-170)

## 7.5. Video resolution and setup

The GAL protocol defines the following video resolutions:

Resolution	Max Video Bit Rate (kbits/s)	Max Frame Rate
800x480	4,000	60 FPS
1280x720	6,000	60 FPS
1920x1080	8,000	60 FPS

All MDs support 800x480 resolution; some high performance MDs support 1280x720 and 1920x1080 resolution as well.

The MD and HU negotiate video resolution during the service discovery:

1. The MD requests the VideoConfigurations supported by the HU.
2. After establishing the video channel, the HU **MUST** send a Config Message with a prioritized list of indices (most preferred first) for the previously-provided VideoConfigurations (R07-185).
3. MD selects a configuration based on HU preference and MD hardware encoder capabilities.
4. MD sends a Start message (with the index of the selected configuration) to HU.

To ensure a good user experience and correct display and rendering of the projected UI, the HU:

- **MUST** accurately report all fields in VideoConfiguration (R07-190)
- **MUST** report the `real_density` as defined in the protocol specification (R07-195).
- **MUST** accurately report the pixel aspect ratio of their screen (R07-200). For displays with non-square pixels, the MD applies correction on the picture sent to the HU to compensate

## 7.6. UI resolution

HU displays rarely have standard physical sizes or match AAP-supported resolutions or aspect ratios, making a 1:1 rendering of the projection UI to the HU display impractical.

To support a wide range of in-vehicle display hardware, AAP uses a logical *UI resolution* that is always less than or equal to the codec resolution of the transported video stream. MDs render content only in the area specified by the UI resolution, with horizontal and vertical margins padded to match the logical UI resolution.

When the MD encodes a video stream, it fills horizontal and vertical margins with arbitrary content. When the HU decodes the stream it MUST clip the video stream to the UI resolution size and discard the margins (R07-210).

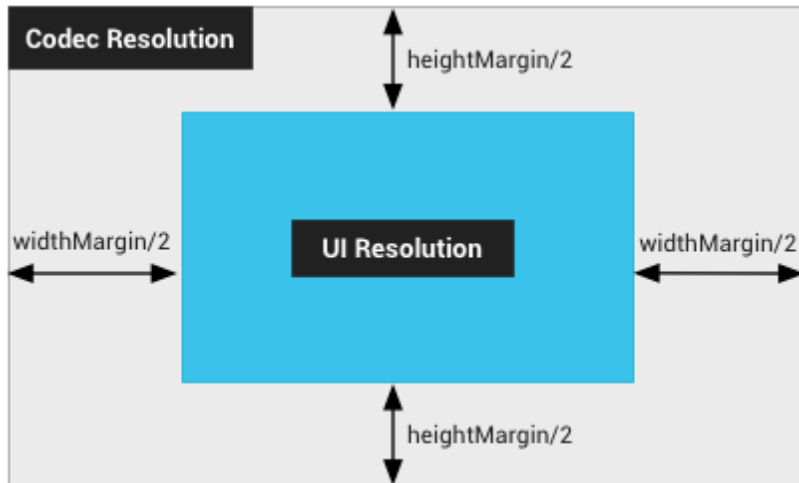


Figure 7.4. Codec resolution and UI resolution

The MD may not completely fill the UI resolution area with visible UI and instead place black bars at the top and bottom (letterbox) or at the sides (pillarbox). Integrations MUST NOT attempt to account and hide any such bars as they may change over time (R07-220).

### 7.6.1. Resolution support

The HU:

- MUST support the 800x480 resolution (R07-230)
- MUST support the 1280x720 resolution only if the width is greater than 800 pixels or the height is greater than 480 pixels (R07-240)
- MUST support the 1920x1080 resolution only if the width is greater than 1280 pixels or the height is greater than 720 pixels (R07-250)

When displaying lower video codec on a higher resolution screen, content MUST be up-scaled to fill the physical screen (or content area) while maintaining aspect ratio (R07-260). Scaling factor MUST NOT be greater than 2.0 (R07-270).

For example, for a screen with a resolution of 1200x720, the HU advertises support for both 1280x720 and 800x480 video. The width margin is set to 80 for the 1280x720 configuration, while 800x480 video is scaled 1.5 times before rendering.

### 7.6.2. Reported screen density

Each VideoConfiguration contains a density. Every screen has a physical density expressed as the number of physical pixels in each inch, known as dots per inch (DPI). The [Screen Size Calculator](#) MUST be used to calculate the reported density (R07-280). The reported density by the HU during video setup:

- MUST be added in the **density** field defined in the protocol specification (R07-281)
- MUST be calculated separately for each video resolution supported by the HU (R07-282)
- MUST compensate for the scaling factor by setting the **Video scaling factor** value in the screen size

- MUST be the value resulted in the **Scaled Density** field in the calculator (R07-284)

The reported density from the Screen Size Calculator optimizes the Android Auto UI (in density-independent pixels), font size for maximum legibility given a viewing distance, and tap target size. It does this while minimizing pillarboxing or letterboxing of the UI inside the video stream.

### 7.6.3. Screen configuration example

This example explains how to determine the correct display settings for a blended UI that displays AAP content next to native content.

#### 7.6.3.1. Screen details

- Screen is 11" diagonal, 1280x600px
- Projection Area is 8.25" diagonal, 960x500px
- Pixel aspect ratio is 1.0 (perfectly square)

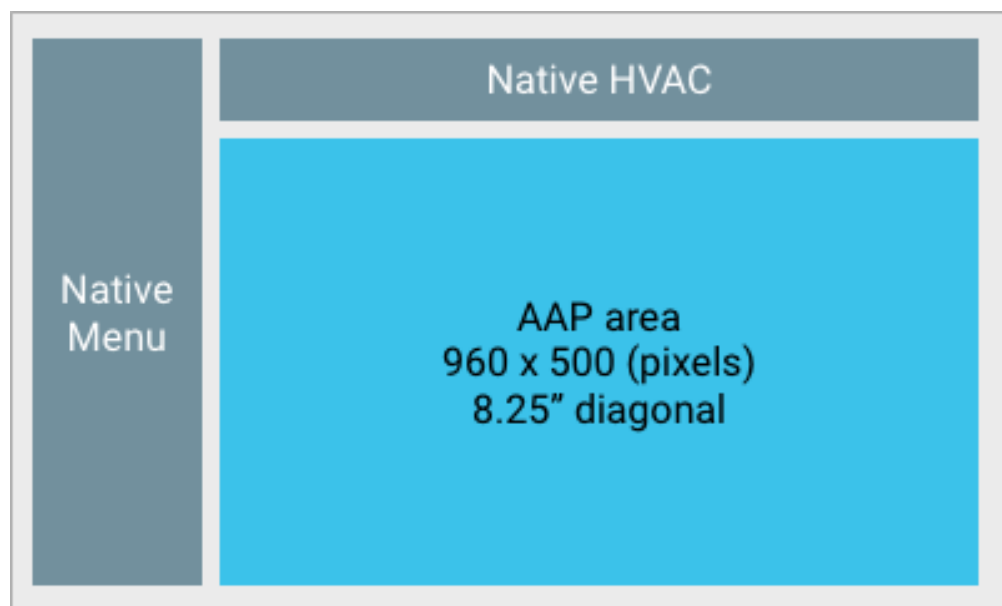


Figure 7.6. Blended HMI layout

#### 7.6.3.2. Screen configuration for 800x480 video

Upscaling is required to fill 960 px width.  $960/800 = 1.2x$  upscale. A vertical margin must be declared by the HU during video setup as the AAP area has a wider aspect ratio than the 800x480px source video.  $500px/1.2x$  upscale = 417 px UI in 480P source video. Vertical margin =  $480px - 417px = 63px$ , as declared in service discovery.

#### 7.6.3.3. Using the Screen Size Calculator

Enter the physical characteristics of the AAP screen area only (i.e., not the entire display if using a blended HMI), including any up-scaling.

## Physical Details

Screen physical diagonal size	<input type="text" value="8.25"/> in
Number of pixels	<input type="text" value="960"/> x <input type="text" value="500"/> px
Viewing distance	<input type="text" value="950"/> mm
Pixel aspect ratio (PAR)	<input type="text" value="1"/>
Video scaling factor	<input type="text" value="1.2"/>
Touch screen	<input checked="" type="checkbox"/>

Figure 7.7. Screen Size Calculator settings.

Click Calculate and observe the colored output.

Reported Density (dpi)	Screen Width (dp)	Screen Height (dp)	Secondary Text Size (arcmin)	64x64dp Touch Target Size (mm)
142	1082	563	11.81	11.0
143	1074	559	11.90	11.1
144	1067	556	11.98	11.2
145	1059	552	12.06	11.2
146	1052	548	12.15	11.3
147	1045	544	12.23	11.4
148	1038	541	12.31	11.5
149	1031	537	12.40	11.5
150	1024	533	12.48	11.6
151	1017	530	12.56	11.7
152	1011	526	12.65	11.8
153	1004	523	12.73	11.8
154	997	519	12.81	11.9
155	991	516	12.89	12.0
156	985	513	12.98	12.1
157	978	510	13.06	12.2
158	972	506	13.14	12.2
159	966	503	13.23	12.3
<b>160</b>	<b>960</b>	<b>500</b>	<b>13.31</b>	<b>12.4</b>
161	954	497	13.39	12.5
162	948	494	13.48	12.5

Figure 7.8. Screen sizes by density.

In this example, the UI size minimum requirements could be fulfilled with a reported density as low as 150 dpi (before factoring in scaling). However, check the suggested **Reported Density** and **Scaled Density** results:

**Suggested reported density**      Reported  ppi (  X  )  
    Scaled  ppi

Figure 7.9. Suggested reported and scaled density values.

- The **reported** value is NOT 150 dpi, even though the results suggest that would be acceptable. This is because Android app developers target specific representative screen density values, and the Screen Size Calculator suggests the *reported* density at one of these specific values (120, 160, 213, 240, etc.) where possible
- The density reported by the HU during video setup will be 133 (not 160) to account for the scaling performed by the HU.

## 7.7. Video characteristics

To preserve bandwidth and power, the AAP protocol sends video frames only for projection UI updates (and does not send frames when the UI is unchanged). HU video decoders MUST NOT enter power-saving mode even when not receiving frames (to avoid adversely affecting latency) (R07-290).

Restricted  
H.264 video sent from MD contains only I frames and P frames (buffering set to minimum). HU SHOULD be able to decode each frame without waiting for additional frames (R07-300). When this is not possible, the HU MUST accurately report the depth of its decoder pipeline as part of service discovery so the MD can send enough idle frames to ensure the display updates properly (R07-310). A larger decoder depth values results in an increase in MD power consumption and can result in increasing latency.

## 7.8. Video latency and flow control

### 7.8.1. Flow control

AAP video streams are real-time, enabling the HU to render streams immediately. Streams use an n-ACK based flow control scheme to enable the video source to throttle throughput when under load. The HU:

- MUST ACK all video packets sent to it (R07-460)
- MUST ACK the video packets as soon as it is able to receive one or more additional packets (R07-465)
- MUST NOT discard video data sent to it (R07-468)

### 7.8.2. Video latency

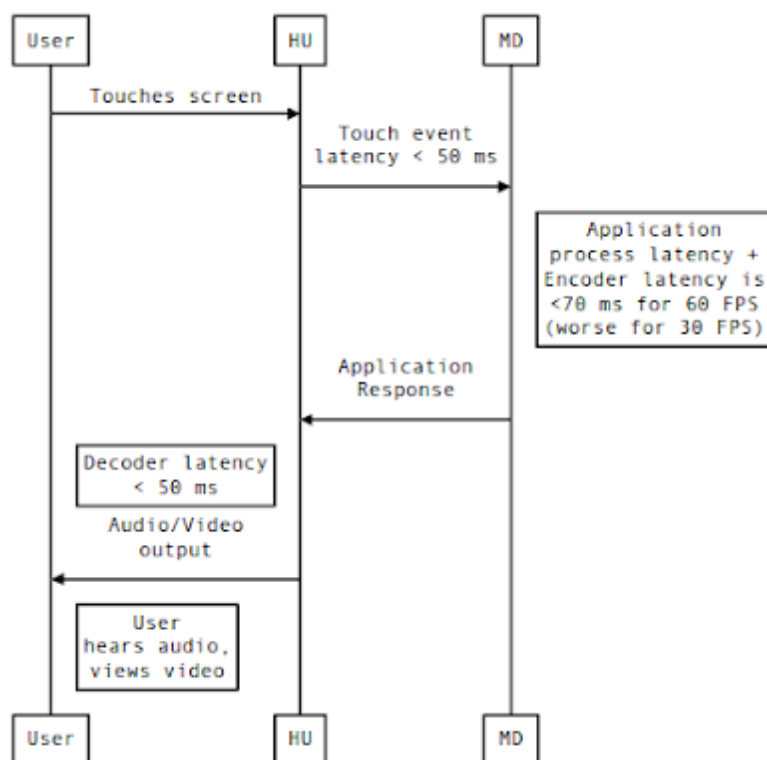
Minimizing two (2) separate video latencies is important to preserve the user experience in AAP: video setup latency and video output latency.

Video *setup latency* is the time required by the vehicle to establish the video stream and includes the time necessary to perform any on-screen transition to projection mode and power on the primary display. Video setup latency SHOULD be under 500ms (R07-470).

Video *output latency* is the delay between a video frame being made available from the AAP receiver and reproduction of the video on the primary display. Output latency includes video buffering performed by the in-vehicle graphics pipeline, video codec decoder latency, and display latency. User-visible video latency is higher than video output latency because video output latency does not include encoder and transport delays. Video output latency SHOULD be under 50ms (R07-480).

The following diagram shows combined latency from user input to UI change. Overall response time MUST be less than 250ms (R07-490). This requirement is tested in [performance testing](#).

### Combined latency





## 7.9. Android Auto UI stream types

HU implementations will receive different UI video streams depending on the display resolution, aspect ratio and input methods.

The HU will receive the widescreen UI if:

- 720p or 1080p video resolution is supported and in use.
- Touchpad or touchscreen input is supported, and a rotary controller is not present.
- The HU fulfills the constraint defined for the respective resolution
  - 720p:  $(1280 - \text{width\_margin}) * 160 / \text{density} \geq 1240$
  - 1080p:  $(1920 - \text{width\_margin}) * 160 / \text{density} \geq 1240$

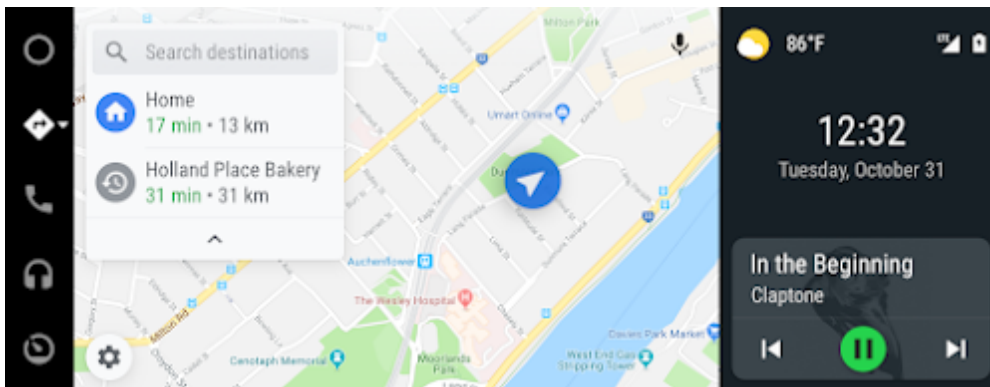


Figure 7.12. Widescreen projection UI.

The HU will receive the normal projection UI with black bars on either horizontal side if:

- 720p or 1080p video resolution is supported and in use.
- A rotary controller is present.
- The HU fulfills the constraint defined for the respective resolution
  - 720p:  $(1280 - \text{width\_margin}) * 160 / \text{density} \geq 1240$
  - 1080p:  $(1920 - \text{width\_margin}) * 160 / \text{density} \geq 1240$

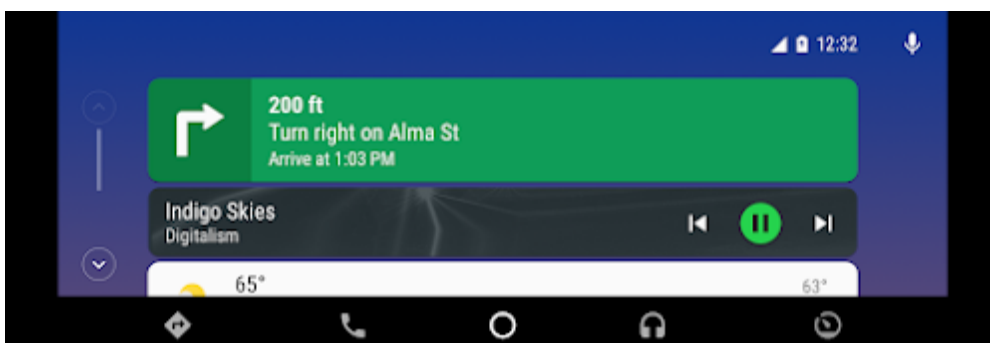


Figure 7.13. Normal projection UI with black bars.

The HU will receive the normal projection UI if:

- The HU fulfills the constraint defined for the respective resolution
  - 720p:  $(1280 - \text{width\_margin}) * 160 / \text{density} < 1240$
  - 1080p:  $(1920 - \text{width\_margin}) * 160 / \text{density} < 1240$





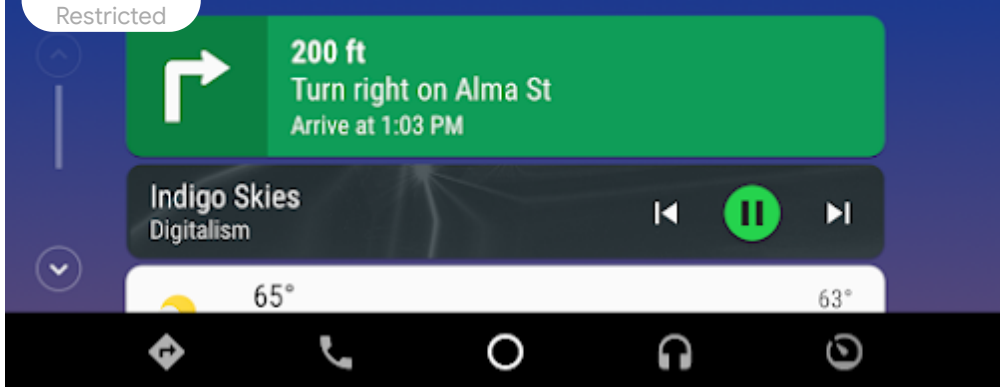


Figure 7.14. Normal projection UI.

where `width_margin` is the number of pixels in the X axis in which content will not render and `density` is the DPI value generated by the screen size calculator for the respective resolution.

## 7.10. Video focus

AAP enables users to seamlessly switch back and forth between the native UI and projection UI, requiring a robust video focus negotiation model. However, because the main console display is also used for safety-critical features (such as a backup camera), the HU **always controls video focus** and determines when the MD can project content to the screen.

AAP includes two (2) main focus states, *Projected* (`VIDEO_FOCUS_PROJECTED`) and *Native* (`VIDEO_FOCUS_NATIVE`). For transient native screens (such as turn-signal activated blind spot cameras expected to show for a short time only) a third focus state is available: *Native Transient* (`VIDEO_FOCUS_NATIVE_TRANSIENT`). From the HU perspective, Native Transient is the same as Native; the purpose is to provide behavioral hints to the MD. Transient native content (popups/overlays) MAY overlay Projection video (R07-320). For details, see [Overlays](#).

During connection setup, video focus MUST NOT be granted to the MD by the HU until `VideoSink::setupCallback()` has been called (R07-330). Attempting to grant video focus to the MD before or during `VideoSink::setupCallback()` will cause a failure in the VideoSink.

Video Focus State	MD sends video	HU sends touch/controller events
Projected	Y	Y
Native	N	N
Native Transient	N	N
Projected with Native Overlay	Y	Y/N (see <a href="#">Overlays</a> )

### 7.10.1. Projected video focus

When video focus is set to Projected, the HU:

- MUST render the video stream from the MD (R07-340)
- MUST forward all touch/controller events originating in the projected video area to the MD (R07-350)

### 7.10.2. Native video focus

When video focus is set to Native, the HU:

- MUST render the native UI (R07-360)

When the HU is showing a backup camera, video focus MUST be set to Native (R07-375).

When video focus switches from Projection to Native, the HU MAY ignore any additional frames the MD sends from

When the MD switches from Projection to Native, the HU MAY ignore any additional frames the MD sends from the media session that was providing projected video (R07-380). When video focus is set to Native, the MD does not stream video to the HU.

### 7.10.3. Native transient video focus

HU SHOULD use Native Transient video focus state when native video focus is anticipated to only last a short amount of time (R07-390). For example, a turn-signal activated blind spot camera is expected to show for a short time only. Transitioning directly from Native Transient video focus state to Native video focus state is supported.

### 7.10.4. Overlays

HUs MAY display short, transient overlays (such as volume displays) on top of projected video content (R07-400). Unless the overlay is safety critical, it MUST NOT obscure more than 30% of the AAP display area and MUST allow input events from the non-obscured area (R07-410). When a safety critical overlay is being shown, input events MUST NOT be sent to Android Auto from anywhere on the screen (R07-420). There MUST be clear delineation between AAP content and the native overlay (R07-421). For example, a native overlay MUST NOT use gradients or reduce intensity to give the appearance of translucency (scrims).

### 7.10.5. Video focus requests and notifications

The MD will send video focus requests to the HU to request video focus be granted to projection or native. For example, the MD will send a request for projected video focus when receiving a KEYCODE\_SEARCH, or will send a request for native video focus when a user selects the **Return to Native** menu option shown in the AAP UI on the fifth facet.

The HU MUST grant video focus in response to a request from the MD unless it is unsafe to do so (R07-430), such as when the vehicle is in reverse and is showing the backup camera. The MD sends video focus requests only in response to a direct user action such as a key input. The HU MUST NOT take video focus away from AAP except when the user explicitly chooses to return to the native UI or for safety reasons (R07-431).

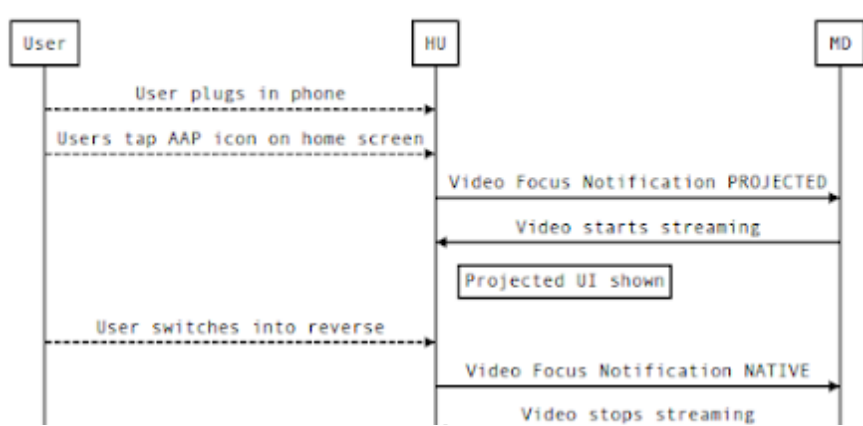
In response to a video focus request, the HU MUST send a video focus notification to the MD, indicating the new video focus state, even if the video focus state remains unchanged (R07-440).

In addition to responding to video focus requests, the HU MUST also send unsolicited video focus notifications based on user actions that change video focus (R07-450). For example, when a user currently viewing Google Maps within projected presses the RADIO hard button on an HU to show the native radio screen, the HU sends a video focus notification (focus = native) to the MD. If the user then selects a new radio station, goes to native home screen, and taps the Android Auto icon, the HU sends another video focus notification (focus = projected), and the projected screen showing Google Maps is shown to the user again.

After establishing an AAP connection, including the MD announcing the video encoder is ready (listen for `IVideoSinkCallbacks::setupCallback`), video focus is set to native. After this point, the MD may request video focus at any time.

### 7.10.6. Video focus examples

#### User Starts AAP and Reverses Out of Their Driveway



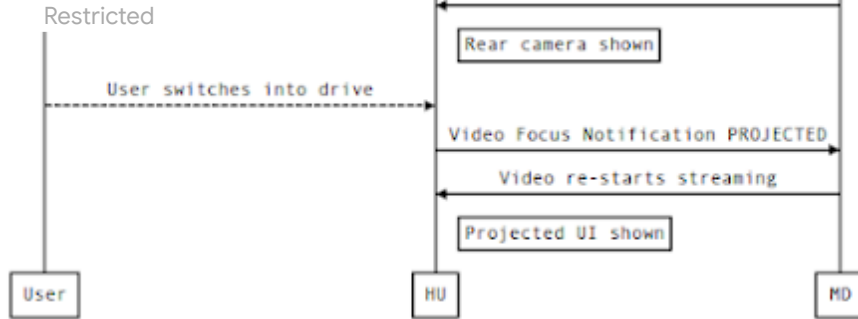


Figure 7.10. User starts AAP and reverses out of driveway.

### User Starts VR Session on Native Home Screen

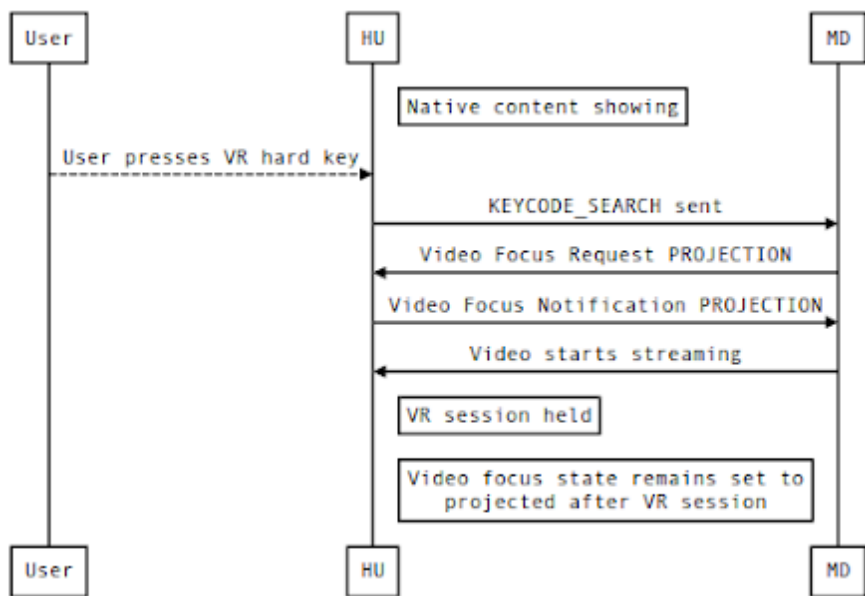


Figure 7.11. User starts VR session on native home screen.

## 7.11. Competing functions

AAP includes several components that mirror functions commonly included in HU native features, e.g. media, navigation, and dialer. In a blended UI or multiple screen implementation, when AAP content is being shown at the same time as the native content, the combined UI **MUST NOT** show competing functions at the same time (R07-451).

When a user places a phone call while in AAP, the AAP dialer facet displays the ongoing call. The HU **MAY** allow the user to switch to the native system and display the native phone call screen during an ongoing call (R07-452). If the user can access the native phone call screen during an ongoing call, the HU **MUST** display the correct call state (R07-453).

If a user is listening to native media, and then switches video focus to AAP, native media must continue playing until the user pushes the play button in the AAP media screen (R07-454).

If native navigation is active and the user starts a route in AAP resulting in the MD requesting navigation focus, the HU **MUST** stop native navigation (R07-455); for details, see [Navigation](#). If native navigation is active, and the user switches video focus to AAP, native navigation **MUST NOT** cancel before the user initiates a route in AAP (R07-456).

## 8. Audio

This section describes AAP audio architecture and audio focus management.

### 8.1. Terminology

- <sup>Restricted</sup> **Audio source (producer).** Logical endpoint that generates audio data and sends to an audio sink. (e.g., native radio, media channel from the MD, guidance channel from the MD, in-vehicle microphone).
- **Audio sink (consumer).** Logical endpoint that receives audio content from an audio source (e.g., in-vehicle speakers, the MD channel receiving microphone captured audio).
- **Audio stream.** Named combination of an audio source and audio sink.

## 8.2. Audio streams

*Native audio streams* are sent from the HU to the HU; *Output audio streams* are sent from the MD to the HU; *Input audio streams* are sent from the HU to the MD.

AAP defines the following six streams (of which two are currently not used).

Stream	Direction	Transport	Purpose	Protocol	Wireless
Guidance	device-to-vehicle	AAP	driver guidance (ASR and turn-by-turn)	PCM 16 bit, 16 kHz, mono.	PCM 16 bit/AAC-LC (depending on what MD selects)
Media	device-to-vehicle	AAP	cabin media	PCM 16 bit, 48 kHz stereo	PCM 16 bit/AAC-LC (depending on what MD selects)
Microphone	vehicle-to-device	AAP	Microphone input	PCM 16 bit, 16 kHz, mono	-
Legacy In-call	bi-directional	Bluetooth SCO	Phone call audio	HFP	-
UI	device-to-vehicle	AAP (future)	user interface feedback	-	-
Voice	device-to-vehicle	AAP (future)	Phone call audio	-	-

Previous versions of the HUIG permitted the HU to support only the AAC-LC codec (instead of the PCM codec). New HU implementations that support only wired AAP MUST support only the PCM codec (R08-010).

For HU implementations that support AAW, the HU MUST support both the PCM codec and AAC-LC codec (R08-012). To do this, the HU registers two audio sinks per device-to-vehicle audio channel, one for PCM and one for AAC-LC.

### 8.2.1. Guidance stream

The guidance output audio stream plays high-priority transient audio such as spoken turn-by-turn navigation instructions, traffic warnings, and Automatic speech recognition (ASR) responses. The HU MUST implement the *Guidance stream (R08-015)*

Guidance stream audio:

- MAY be routed to audio channels aimed at the driver (R08-020)

### 8.2.2. Media stream

The media output audio stream, or AAP Media, plays low-priority, long-form audio from applications on the MD such as music, audiobooks, podcasts, Internet radio, etc. The HU MUST implement the Media stream (R08-025).

Native media is a native audio stream that plays low-priority, long-form audio from and to the HU, such as native radio.

Media stream audio:

- MUST be routed to the main cabin audio sink (R08-030)
- MUST NOT play on top of any other audio stream (other streams cannot duck under this stream) (R08-040)
- MUST be mutually exclusive with Native media (R08-050)

### 8.2.3. Microphone

The microphone input audio stream captures microphone audio using the built-in vehicle microphone(s) and transmits that audio to the MD, which can use it for ASR, etc. For additional microphone requirements, see [Automatic Speech Recognition](#).

### 8.2.4. Legacy in-call audio

In-call voice audio is currently routed over a Bluetooth HFP/SCO link instead of the AAP protocol. Negotiation of the voice call audio link is performed using Bluetooth HFP and is not detailed in this guide.

### 8.2.5. UI stream

AAP does not currently use the UI stream. The HU MUST NOT implement the UI stream (R08-060). The UI feedback output audio stream is intended for short audio feedback (beeps, chimes, etc.) in response to UI input. The MD does not send focus requests for this stream.

### 8.2.6. Voice stream

AAP does not currently use the voice stream and implementation is not yet possible. The HU MUST NOT implement the Voice Stream (R08-065). The voice stream is designed to handle telephony and the voice output audio stream plays conversational voice audio such as voice call output.

## 8.3. Audio focus

Audio focus is relevant for the Guidance and Media streams. The MD will not send audio on these streams without first requesting and obtaining audio focus. Audio focus **requests** inform the HU the MD wants audio focus. An MD sends a request when it wants to play audio. Audio focus **notifications** inform the MD of the current audio focus state. The HU sends a notification in response to audio focus requests or unsolicited (prompted by external events such as the user selecting a native audio source).

Integrators MUST implement audio focus handling to provide a seamless user experience for audio from the MD and HU (R08-070).

Audio focus is **completely** separate from video focus. The HU:

- MUST be able to have video focus while the MD has audio focus (R08-080)
- MUST be able to have audio focus while the MD has video focus (R08-090)
- MUST be able to switch audio focus without switching video focus (R08-100)
- MUST be able to switch video focus without switching audio focus (R08-110)

### 8.3.1. Audio focus requests

Audio focus requests are sent from the MD to the HU. Android Auto supports four (4) types of audio focus requests.

Type	Description	Permitted Responses
GAIN	Request to gain audio focus for unknown duration.	STATE_GAIN STATE_LOSS STATE_LOSS_TRANSIENT STATE_LOSS_TRANSIENT_CAN_DUCK
GAIN_TRANSIENT	Request to gain audio focus for a short duration. The MD sends this request only when it does not already have full (GAIN) focus.	STATE_GAIN STATE_GAIN_TRANSIENT STATE_LOSS STATE_LOSS_TRANSIENT STATE_LOSS_TRANSIENT_CAN_DUCK
GAIN_TRANSIENT_MAY_DUCK	Request to gain audio focus for a short duration while indicating that it is acceptable for other audio sources to keep playing after having lowered their output level (referred to as <i>ducking</i> ). The MD sends this request only when it does not already have STATE_GAIN or STATE_GAIN_TRANSIENT audio focus.	STATE_GAIN STATE_GAIN_TRANSIENT STATE_GAIN_TRANSIENT_GUIDANCE_ONLY STATE_LOSS STATE_LOSS_TRANSIENT STATE_LOSS_TRANSIENT_CAN_DUCK
RELEASE	Request to release audio focus.	STATE_LOSS

The HU:

- MUST support audio focus GAIN requests (R08-115)
- MUST support audio focus GAIN\_TRANSIENT requests (R08-116)
- MUST support audio focus GAIN\_TRANSIENT\_MAY\_DUCK requests (R08-117)
- MUST support audio focus RELEASE requests (R08-118)

### 8.3.2. Audio focus notifications

Audio focus notifications are sent by the HU to the MD to inform the MD of the audio focus state. Audio focus notifications **MUST** be sent in response to audio focus requests (R08-120). When responding to an audio focus request, the audio focus notification sent by the HU **MUST** be a permitted response as per the above table (R08-130). Audio focus notifications **MUST** be sent unsolicited when an external event results in a change in audio focus state (R08-140). For example, the HU sends an audio focus notification of STATE\_LOSS to the MD when the user turns on FM radio.

If an HU wants to play native navigation guidance when the MD has STATE\_GAIN, it **MUST** send a STATE\_LOSS\_TRANSIENT\_CAN\_DUCK audio focus notification (R08-141). The MD will duck its media playback until

Restricted  
TRANSIENT\_CAN\_DUCK audio focus notification (R08-141). The MD will duck its media playback until the HU grants it STATE\_GAIN when the native navigation guidance is over.

If an HU wants to play native notifications when the MD has STATE\_GAIN, it MUST send a STATE\_LOSS\_TRANSIENT or STATE\_LOSS\_TRANSIENT\_CAN\_DUCK audio focus notification (R08-142). The HU MUST NOT send unsolicited STATE\_GAIN\_TRANSIENT or STATE\_GAIN\_TRANSIENT\_GUIDANCE\_ONLY messages (R08-150).

Android Auto supports the following focus states for use in audio focus notifications:

Response	Indicates ...
STATE_GAIN	MD has gained audio focus (when in response to an audio focus request) or regained audio focus (when it was temporarily lost).
STATE_GAIN_TRANSIENT	MD has gained audio focus for a short duration.
STATE_GAIN_TRANSIENT_GUIDANCE_ONLY	MD has gained audio focus for a short duration but is only allowed to play on the GUIDANCE channel. This audio focus state MUST ONLY be sent if the vehicle cannot mix MD media with vehicle media, and then only in response to a GAIN_TRANSIENT_MAY_DUCK request (R08-160).
STATE_GAIN_MEDIA_ONLY	Unused. Was previously provided for use by HUs that cannot mix GUIDANCE and MEDIA audio streams.
STATE_LOSS_TRANSIENT_CAN_DUCK	MD can continue playing through MEDIA channel if volume is lowered.
STATE_LOSS	MD has lost audio focus or failed to gain audio focus.
STATE_LOSS_TRANSIENT	MD has temporarily lost audio focus or has failed to gain audio focus.

### 8.3.3. Audio focus implementation details

If the audio focus request indicates ducking is allowed (GAIN\_TRANSIENT\_MAY\_DUCK), the HU MUST duck audio on native media and mix in transient audio from the MD (R08-170). When the HU is ducking native media and mixing in transient audio from the MD, the HU MUST ensure that transient audio from the MD is clearly audible to the driver (R08-175).

- If the MD wants to play audio on the Media or Guidance stream but the focus state is such that the MD cannot play audio on the desired stream, it will send a focus request. The MD will wait (timeout: 500ms) for an audio focus notification before playing any audio.
- If this timeout expires and the audio focus state remains unchanged, the HU MUST send audio focus notifications in response to audio focus requests in such a manner as to avoid timeouts (R08-180).
- If the HU requires time to process audio focus requests (to switch audio paths, initialize amps, etc.) then the HU should finish the audioFocusRequestCallback immediately without blocking and send its focus notification asynchronously after its required setup has completed.
- If setup will take more than 500ms, the HU should send the audio focus notification earlier (to avoid focus request timeout) and add additional delay to audio stream by buffering audio and postponing ACK when the MD



Restricted but) and add additional delay to audio stream by buffering audio and postponing ACK when the MD starts sending audio stream, until the setup is complete.

Audio focus requests MUST be rejected only for safety reasons (R08-190). Specifically, if an HU has native navigation and is playing native navigation guidance, it MAY (for safety reasons) give precedence to native navigation guidance and reject GAIN\_TRANSIENT\_MAY\_DUCK audio focus requests from the MD (R08-200).

In all situations, the HU makes the final decision on audio focus. HU MAY play urgent audio for safety reasons independent of audio focus state (R08-210). With the exception of safety-critical audio, if the HU does not have audio focus, the HU MUST send an audio focus notification before playing audio from a native audio source (R08-220). This is to enable the MD to synchronize pause and duck timing. If the focus state is unchanged, notification is unnecessary (for example, when switching from listening to a CD to native radio).

After granting audio focus to the MD, the HU MUST play all audio packets sent by the MD (R08-230). After receiving an audio focus loss notification, the MD will stop audio playback (with allowances for short duration audio streaming due to asynchronous communication).

The HU MUST honor the last audio focus state when receiving a RELEASE notification (R08-240). If the MD held STATE\_GAIN and it relinquished audio focus (RELEASE) without a corresponding audio focus request from the HU, the HU MUST NOT automatically resume playing native media (R08-250). If the MD held STATE\_GAIN\_TRANSIENT, STATE\_GAIN\_TRANSIENT\_MAY\_DUCK or STATE\_GAIN\_TRANSIENT\_GUIDANCE\_ONLY and it relinquished audio focus (RELEASE), the HU MUST immediately restore focus to the last active media source (R08-260).

After establishing an AAP connection, the HU has audio focus and the MD will request focus before playing audio.

#### 8.3.4. Audio focus during phone calls

Phone calls (incoming and outgoing) are handled by Bluetooth. When a phone call is made or received, the HU MUST stop playing all native media (R08-310). The HU MAY play transient native audio (such as navigation guidance) while on a call (R08-311). The MD will also pause/stop playback of any media it is playing.

The MD may play audio on the guidance stream (such as navigation guidance) during a call. If it does not have audio focus, the MD will request audio focus with a GAIN\_TRANSIENT, GAIN\_TRANSIENT\_MAY\_DUCK, or other request. If the HU cannot play any audio during a phone call and the MD had audio focus before the phone call, the HU MUST send the MD STATE\_LOSS\_TRANSIENT notification to prevent the MD from playing audio (R08-320). If the MD requests audio focus during a phone call and the HU cannot grant the request, the HU MUST reject the request with a STATE\_LOSS\_TRANSIENT notification (R08-330). For details on handling the audio focus when the phone call ends, see [Audio focus implementation details](#).

#### 8.3.5. Automatic speech recognition (ASR) focus

During a projected ASR session, all native audio sources MUST stop/mute (with the exception of safety critical messages) (R08-270). After a projected ASR session completes, if the HU has audio focus the HU MUST resume/unmute any native media that was playing when the projected ASR session started (R08-280). When a projected ASR session starts, the MD sends a MESSAGE\_VOICE\_SESSION\_NOTIFICATION to notify the HU. If it does not already have audio focus, the MD will request focus before playing any beep tones or ASR responses.

**NOTE:** If the user starts an ASR session when listening to AAP media, the HU receives both a MESSAGE\_VOICE\_SESSION\_NOTIFICATION with status = VOICE\_SESSION\_START and a MEDIA\_MESSAGE\_STOP indicating that media has stopped. After the ASR session has finished, the HU receives a MEDIA\_MESSAGE\_START message and MESSAGE\_VOICE\_SESSION\_NOTIFICATION with status = VOICE\_SESSION\_END. The MEDIA\_MESSAGES and MESSAGE\_VOICE\_SESSION\_NOTIFICATIONs are handled by different asynchronous processes on the MD, and as such their relative order cannot be ensured.

Before starting a native ASR session, if the MD has audio focus, the HU MUST notify the MD that it is temporarily acquiring audio focus (send a STATE\_LOSS\_TRANSIENT audio focus notification before ASR) and then restore the previous focus state after native ASR completes (R08-290).

For details on ASR integration, refer to [Automatic Speech Recognition](#).

### 8.3.6. Audio mixing

Earlier versions of this document included guidance for HUs that cannot mix GUIDANCE and MEDIA audio streams. This guidance has been removed. All HUs MUST support mixing GUIDANCE and MEDIA audio streams (R08-300).

## 8.4. Audio latency and flow control

### 8.4.1. Flow control

AAP audio streams use an n-ACK based flow control scheme. HU MUST ACK all audio packets sent to it (R08-370). The audio source can implement flow control using the ACK messages, including resampling in the case of clock skew. HU SHOULD wait until receiving two audio packets before starting playback to minimize buffer underruns (R08-380).

### 8.4.2. Audio latency

To preserve the user experience in AAP, it is important to minimize two (2) separate audio latencies: audio setup latency and audio output latency.

Audio setup latency is the time required by the HU to establish an audio stream, measured as the maximum time between an audio focus request to the HU and a response from the HU. This latency includes the time necessary to:

- Duck other audio
- Power Digital-to-Analog Converter (DAC) and amplifiers
- Configure HU mixer

Audio setup latency MUST be under 500ms (R08-390).

Audio output latency is the delay between audio being made available from the AAP receiver and reproduction of the audio on in-vehicle speakers. It is measured as the maximum processing delay in the pipeline once the HU has received two audio frames. This latency includes:

- Buffering performed by the in-vehicle audio pipeline
- Audio codec decoder latency

Audio output latency MUST be under 50ms (R08-400).

## 8.5. Audio volume

All AAP audio streams are line-level volume, making the vehicle responsible for applying volume levels. The vehicle does not notify the MD of volume levels and the MD cannot request volume levels. The HU MUST calibrate AAP audio streams so they are equalized to native audio streams (R08-405). The HU MUST NOT adjust the output of the MD audio through ducking or boosting aside from user initiated volume changes (R08-408).

To avoid driver distraction, sound level adjustments for AAP MUST follow the same style and mechanics used to adjust native sound channels (R08-410). The HU MUST allow the user to independently set volumes for a minimum of three (3) streams: Media, Guidance/ASR, and Call (R08-420). The HU MAY allow the user to independently set volumes for ASR and guidance (R08-430). The HU MUST maintain/remember volume settings for each stream and not reset them between Android Auto sessions (R08-435).

When volume is being adjusted, the HU MUST display the volume (R08-440). When using the steering wheel controls to change the volume, the HU may display the volume bar on the cluster screen instead of the main screen. The volume being adjusted MUST be labelled with a stream-specific text string or icon (R08-450). If AAP is being displayed full-screen, volume MUST be shown in an overlay (R08-460).

If the HU has a mute control, it MUST implement mute of AAP functionality as either Global Mute or Source-Specific Mute (R08-470).

### 8.5.1. Global Mute

In Global Mute, all audio sources are muted. If the HU implements Global Mute it MUST NOT play any audio from the MD when muted, except phone call Audio (R08-480). The HU MUST remain in the mute state after a phone call ends (R08-485). When the MD sends audio while the HU is in Global Mute state, the HU MUST inform the user that audio is muted (R08-490). When in Global Mute, the HU MUST grant all audio focus requests but remain muted (R08-500).

### 8.5.2. Source-Specific Mute

In Source-Specific Mute, mute is managed per audio source. If the HU implements source-specific mute, when the user triggers mute, the HU:

- MUST mute the current audio source (R08-510)
- MUST NOT mute guidance/notifications from the MD (R08-520)
- MAY mute incoming call ringtone audio (R08-530)
- MUST NOT mute phone call audio (R08-540)
- MUST NOT mute audio within an ASR session (which includes readout of received messages) (R08-550)

If the HU implements source-specific mute and the current source is AAP media, the HU MAY stop AAP media playback when the HU is muted; for example, the HU MAY send KEYCODE\_MEDIA\_PAUSE to the MD to pause AAP media and KEYCODE\_MEDIA\_PLAY to resume AAP media (R08-560). Any affordance that implements this functionality SHOULD have a "mute/pause" (or equivalent) label (R08-561). When in Source-Specific Mute, the HU MUST NOT unmute until it is user initiated (for example, the HU MUST NOT unmute media after receiving guidance from the MD) (R08-570).

## 8.6. Audio focus examples

This section gives several specific examples of audio focus requests and notifications being used in practice.

### User Enables Google Play Music

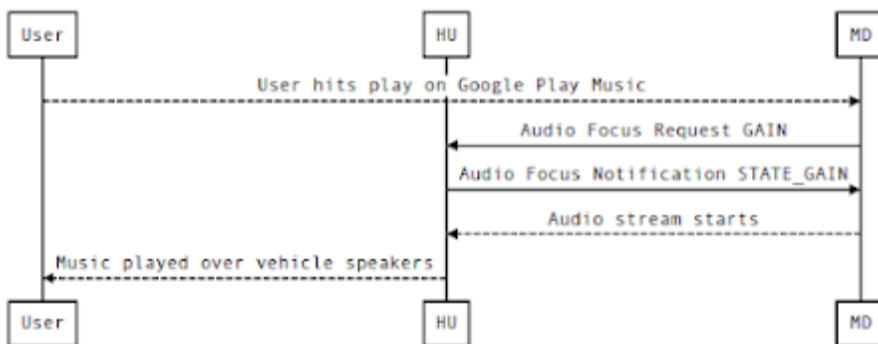
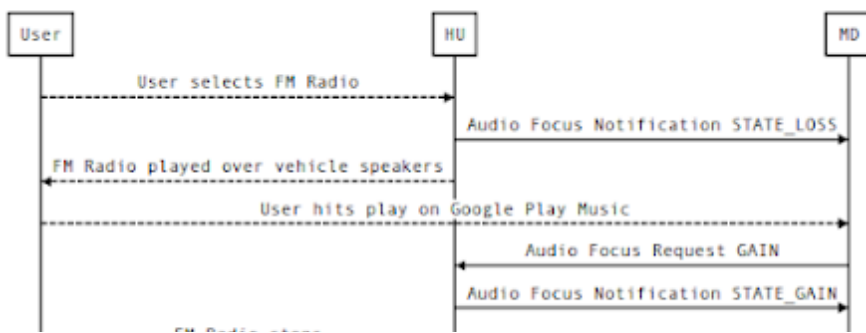


Figure 8.1. User enables Google Play Music.

### User Enables FM Radio Then Google Play Music

#### User Enables FM Radio Then Google Play Music



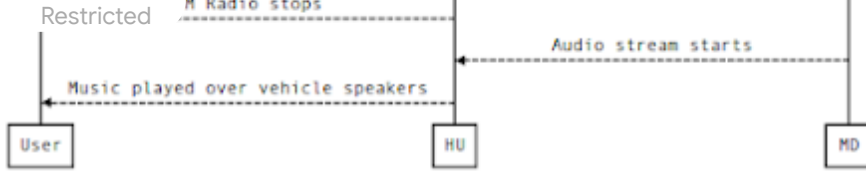


Figure 8.2. User enables FM radio then Google Play Music.

## User Enables FM Radio Then MD Navigation Guidance

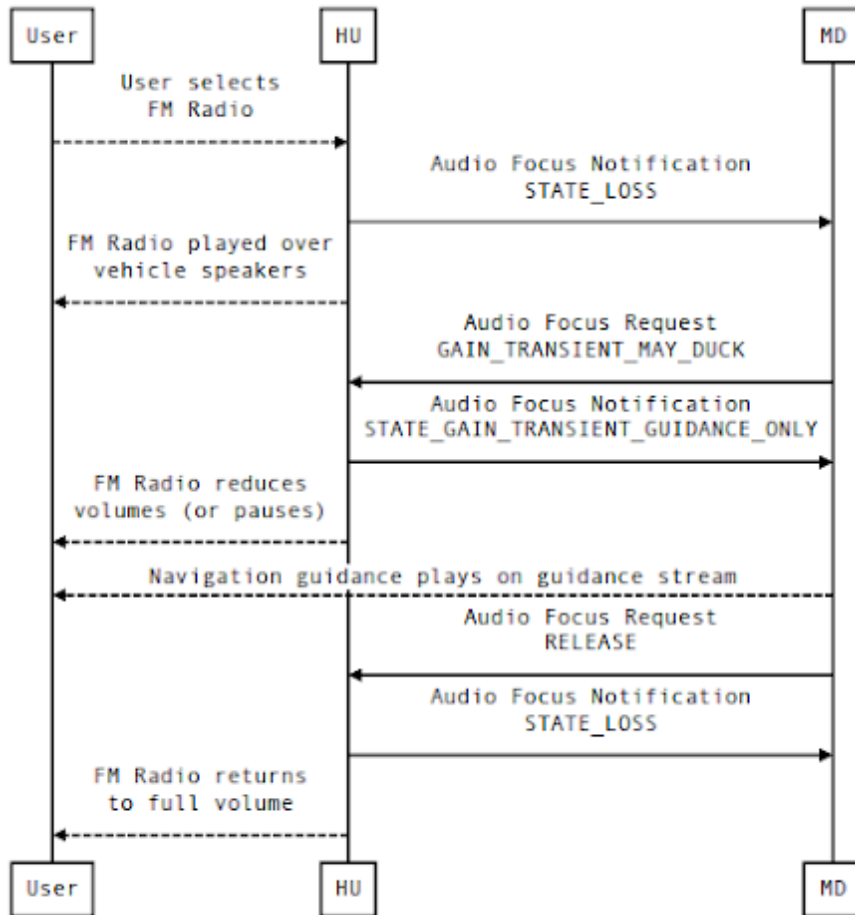


Figure 8.3. User enables FM radio then MD navigation guidance.

In this diagram, the HU cannot mix two MEDIA streams, and hence responds to the GAIN\_TRANSIENT\_MAY\_DUCK audio focus request with a STATE\_GAIN\_TRANSIENT\_GUIDANCE\_ONLY audio focus notification to ensure that audio from the MD is sent over the guidance stream only.

## 9. Input devices

The HU passes input events from the on-board HMI to the MD. AAP supports the following input device types:

- Touchscreen buttons (including D-Pads)
- Touchpad
- Dials/rotary controllers
- Hardware buttons/inputs
- Software buttons/inputs

To provide access to an input device, OS Adaptation Layer implementations MUST register the input device with the AAP receiver when projection starts and report input events to the AAP receiver during projection for key events requested by the MD (R09-005).

The AAP receiver provides the InputSource interface that handles input device registration and event messaging for

Restricted  
The AAP provides the input/output interface that handles input device registration and event messaging for all input device types (for details, refer to the [AAP Receiver Library documentation](#)).

The HU MUST declare one of the following supported input device combinations in Service Discovery (R09-010):

- Touchscreen only
- Rotary only
- Touchpad with the `ui_navigation` flag set to true
- Touchscreen and rotary
- Rotary and touchpad with the `ui_navigation` flag set to false
- Touchscreen and touchpad

AAP has different requirements based on the input(s) supported in the vehicle.

## 9.1. Touchscreen

AAP limits input events to a single touchscreen. When video focus is Projection, the HU MUST send all touchscreen events to the AAP receiver (R09-020).

## 9.2. Touchpad

If an HU supports a touchpad, it MUST declare a touchpad as an input device (R09-026).

If an HU supports multiple touchpads, it MUST declare only one touchpad and combine multiple touchpad inputs on the HU side (R09-022).

When registering a touchpad, the HU MAY set the `ui_navigation` flag to true (R09-029). When a touchpad is registered with the `ui_navigation` flag set to true, the touchpad can be used to navigate the AAP UI.

When registering a touchpad, the HU MAY set the `ui_navigation` flag to false (R09-027). When a touchpad is registered with the `ui_navigation` flag set to false, the touchpad can be used for handwritten text input and map scrolling. When a touchpad is registered with the `ui_navigation` flag set to false, the HU MUST also register either touchscreen or rotary controller (R09-028).

If an HU registers a touchpad for AAP, it MUST send all touchpad events with absolute position values to the AAP receiver when video focus is Projection (R09-030). Values for multiple fingers MAY be sent (R09-040).

### 9.2.1. Touchpad feedback

If an HU registers a touchpad, it MAY support haptic and/or acoustic feedback for touchpad as a response to the below feedback events from an MD (R09-031). Refer to `FeedbackEvent` in the protocol specification for more details.

Event	Status
FEEDBACK_SELECT	Supported
FEEDBACK_FOCUS_CHANGE	Supported
FEEDBACK_DRAG_SELECT	Not supported
FEEDBACK_DRAG_START	Not supported
Event	Status

Restricted_DRAG_END	Not supported
---------------------	---------------

### 9.2.2. Touchpad sensitivity

An HU MUST specify the sensitivity of a touchpad during service discovery (R09-041). An HU MAY update its sensitivity while an AAP session is active by calling ReceiverLib function `InputSource::updateTouchPadUiSensitivity()` with the new sensitivity setting as its parameter and then calling `GalReceiver::updateService()` with the InputSource object as its parameter (R09-042). An HU MUST specify a sensitivity setting between 1 and 10 (R09-045). The default sensitivity setting is 5. The more sensitive a touchpad is configured, the smaller amount a user needs to move their finger over the touchpad to trigger movement.

### 9.2.3. Mechanical click

If an HU registers a touchpad, it MUST have a mechanical click mechanism on the touchpad to trigger the select action (R09-050). The HU MUST send a KEYCODE\_DPAD\_CENTER down event to the MD when it detects a mechanical click gesture has started (R09-056). The HU MUST send a corresponding KEYCODE\_DPAD\_CENTER up event to the MD when it detects a mechanical click gesture has completed (R09-057). The HU MUST send one KEYCODE\_DPAD\_CENTER up event for every KEYCODE\_DPAD\_CENTER down event (R09-058).

## 9.3. Rotary input

AAP supports rotational input devices (typically physical dials associated with the main HU). When projection mode begins, all absolute and relative value device events that impact the main console display MUST register with the AAP receiver (R09-060).

Motion of a rotary controller is modeled as a series of keycodes, with a keypress being sent for each step the controller is moved.

- If the HU supports rotary, the rotary controller MUST support middle select, clockwise rotation (CW), and counterclockwise rotation (CCW) (R09-070).
- If the HU does NOT have a touchpad, the HU MUST have a rotary controller and the rotary controller MUST be able to report clockwise rotation (CW), counterclockwise rotation (CCW), up, down, right, left, and middle select (R09-080).
- If the HU has a touchscreen and rotary, the rotary controller MAY support up, down, right, and left (R09-090).
- If the rotary controller does NOT support up, down, right, and left, a back button MUST be supported (R09-100).

An HU MUST NOT register a rotary controller and a touchpad with the `ui_navigation` flag set to true at the same time (R09-065).

### 9.3.1. Rotary keycodes

Event	Keycode	Action
Tilt Right	KEYCODE_DPAD_RIGHT	Sends DPAD_RIGHT event to projected UI.
Tilt Left	KEYCODE_DPAD_LEFT	Sends DPAD_LEFT event to projected UI.
Tilt Up	KEYCODE_DPAD_UP	Sends DPAD_UP event to projected UI.
Tilt Down	KEYCODE_DPAD_DOWN	Sends DPAD_DOWN event to projected UI.



Restricted Middle Select (enter)	KEYCODE_DPAD_CENTER	Sends KEYCODE_DPAD_CENTER event to projected UI.
Rotation	KEYCODE_ROTARY_CONTROLLER	Special keycode used as a device identifier and reported during initialization (not sent as normal keycode). Rotary controller events are sent using relativeEvent API calls. The delta value SHOULD be positive for clockwise rotation and SHOULD be negative for counterclockwise rotation. An absolute value of one corresponds to a single step in the UI and scales linearly for larger absolute values.

## 9.4. Hardware input actions

Hardware inputs are any non-screen-based surface, switch, knob, rocker, or button used to control vehicle functionality. The Vehicle MUST provide a hardware input for volume control (R09-110). The Vehicle MUST provide a hardware input for Automatic Speech Recognition (R09-120).

If the HU has physical buttons that map to AAP functionality, the implementation MUST meet the [Keycode rules](#) (R09-130).

To provide access to physical buttons, OS Adaptation Layer implementations MUST register each button with the AAP receiver and report key events to the receiver. When projection mode begins, key events available to the MD during projection mode MUST register with the AAP receiver. Button events that do not impact the main console display (window defrost, hazard indicators, etc.) do not need to be sent through AAP.

## 9.5. Software input actions

Software input actions are any graphical user interface (GUI) element used to launch an activity. Software inputs can be one of the following:

- *Persistent*. On-screen affordance that remains in the same position and maintains the same functionality mapping at all times, similar to a software-hardware button.
- *Contextual*. Any control that activates different functionality depending on the context or state of the interface and the vehicle.
- *Modal*. A menu that displays content that temporarily blocks interactions with the main view of the UI.

If the HU has software inputs that map to AAP functionality, the implementation MUST meet the [Keycode rules](#) (R09-140).

## 9.6. Handling of input actions

Features such as navigation/maps, media, and telephony MAY be included as software or hardware inputs in AAP. If the HU supports input actions, it MUST meet the following requirements:

- The protocol-supported activities for that component MUST meet the [Keycode rules](#) (R09-145).
- Each affordance mapping MUST clearly match the label on the affordance (R09-150). For example, the telephone icon should map to the telephony facet instead of mapping to the home screen.
- The integration MUST set a common mapping style for all affordances (R09-160). For example, if the navigation key is mapped to AA, the home affordance should also be mapped.

Integrators MAY offer a toggle behavior for protocol-supported activities that exist in the AAP facet bar (R09-170). For example, pressing a navigation affordance can toggle between Native navigation and Projected navigation.



## 9.7. Source input

If a media source switch input is supported, it MUST include AAP as a source (R09-180). All native screens that provide a UI for switching between media sources must include AAP (R09-190).

When the source input is used to select AAP as a media source, the HU MUST either send KEYCODE\_MEDIA (which switches to Android Auto media facet) or send KEYCODE\_MEDIA\_PLAY (which starts Android Auto media but does not cause screen changes) (R09-200).

## 9.8. Keycodes

The vehicle sends key events to the MD through basic KeyEvent and keycodes that specify the button pressed. AAP uses the standard keycode definitions provided in the Android KeyEvent class. The OS Adaptation Layer MUST map all button events to the appropriate KeyEvent keycode before passing to the AAP receiver (R09-205). Both UP and DOWN events MUST be transmitted for each keypress (R09-206).

When video focus is set to Native, the HU MUST continue to send relevant keycode events to the MD (R09-210). If the MD requests video focus when the keycode event is sent the HU MUST grant video focus (R09-211).

Following are keycode events that can be sent to AAP with the corresponding requirements:

- KEYCODE\_MEDIA. SHOULD be mapped to AA and send keycode\_media when initiated by the user (R09-215). If the Media button is the only available affordance to access and switch media sources, the button MAY be mapped to a screen that supports source management (MUST include AAP) instead of directly mapped to Android Auto (R09-216).
- KEYCODE\_TEL. If the vehicle has an affordance for telephone and AAP has videofocus, the telephone affordance MUST be mapped to AAP and send keycode\_tel when initiated by the user (R09-220). If the HU has videofocus the telephone affordance CAN be mapped to AAP and send keycode\_tel when initiated by the user (R09-225).
- KEYCODE\_CALL. SHOULD not be used. The Bluetooth HFP 'ATA' command MUST be used for answering incoming calls (R09-227).
- KEYCODE\_ENDCALL. SHOULD not be used. The Bluetooth HFP 'AT+CHUP' command MUST be used for rejecting incoming calls as well as ending active calls (R09-228).
- KEYCODE\_NAVIGATION. SHOULD be mapped to AA and send keycode\_navigation when initiated by the user (R09-230). MAY be used as a toggle between native navigation and AAP navigation (R09-235).
- KEYCODE\_HOME. SHOULD be mapped to AA and send keycode\_home when initiated by the user (R09-245). MAY be used as a toggle between native home and AAP home (R09-250).
- KEYCODE\_SEARCH. HU MUST have button affordance. HU MUST send keycode\_search when initiated by the user (for details, see [Automatic Speech Recognition](#)) (R09-253).
- KEYCODE\_BACK. If the vehicle has an affordance for back and AAP has video focus, the back affordance MUST be mapped to AAP and send keycode\_back when initiated by the user (R09-255).
- KEYCODE\_MEDIA\_PLAY\_PAUSE. If the HU has a hardkey that plays and pauses media, the HU must send KEYCODE\_MEDIA\_PLAY\_PAUSE to the MD if the button is pushed when AAP has audio focus(R09-260).
- KEYCODE\_MEDIA\_PLAY. If the HU has a hardkey Play button, the HU MUST send KEYCODE\_MEDIA\_PLAY to the MD if the button is pushed when AAP has audio focus (R09-261).
- KEYCODE\_MEDIA\_PAUSE. If the HU has a hardkey Pause button, the HU MUST send KEYCODE\_MEDIA\_Pause to the MD if the button is pushed when AAP has audio focus(R09-262).
- KEYCODE\_MEDIA\_PREVIOUS. If the HU has a hardkey Previous button, the HU MUST send KEYCODE\_MEDIA\_PREVIOUS to the MD if the button is pushed when AAP has audio focus(R09-263).
- KEYCODE\_MEDIA\_NEXT. If the HU has a hardkey Next button, the HU MUST send KEYCODE\_MEDIA\_NEXT to the MD if the button is pushed when AAP has audio focus (R09-264).

## 10. Automatic Speech Recognition

AAP relies on Automatic Speech Recognition (ASR) for safe and convenient use of AAP while driving. Voice is the preferred method of in-vehicle interaction and requires careful AAP integration. When users launch ASR, the system uses the vehicle cabin microphone(s) and passes audio to the MD via the infotainment HU and AAP link. Vehicles **MUST** have a microphone (R10-005). ASR is assumed to be conversational, involving sequential audio input and audio output.

Vehicles **MUST** provide a hardware input that allows a user to initiate an ASR session (R10-010). The hardware input that initiates ASR:

- **SHOULD** be a button dedicated to ASR (R10-020)
- **MAY** be a button shared with other functions, with long-press mapped to ASR (R10-030)
- **SHOULD** be located on the steering wheel (R10-040)

Users can also initiate an ASR session using the software affordance included in the AAP UI and hotword detection ("OK Google"). Hotword detection is performed by the MD, and as such the vehicle **SHOULD** specify a placement location for the MD where it can pick up hotwords spoken in the cabin (R10-050).

### 10.1. Native ASR

Often in addition to ASR provided by AAP, HUs have native ASR for use cases involving native functionality such as "Turn on radio". HUs **MAY** provide access to native ASR during active AAP connections (R10-060). Native ASR sessions **MUST NOT** display content in an overlay over AAP content; such sessions **MUST** switch video focus to native (R10-070).

### 10.2. Initiating ASR

AAP supports two (2) methods for initializing an ASR session using a hardware button: short-press and long-press. HU **MUST** initialize an ASR session with either a short-press or a long-press of the hardware button (R10-080). If an HU has native ASR, it **MUST** implement the long-press method (R10-081). If the HU does not have native ASR and the hardware button used to launch AAP ASR is not a dedicated button, the HU **MUST** implement the long-press method (R10-082). If the HU does not have native ASR and has a dedicated hardware button for AAP ASR, the HU **MUST** implement the short-press method (R10-083).

#### 10.2.1. Short-Press

In a short-press implementation:

- The HU **MUST** send the `KEYCODE_SEARCH` keycode using the `reportKey` function as soon as the ASR hardware button is pressed down (R10-090).
- Short-press **MUST** trigger AAP ASR regardless of video focus (R10-110).

#### 10.2.2. Long-Press

In a long-press implementation:

- The HU **MUST** use the `reportKeyLongPress` function to send the `KEYCODE_SEARCH` keycode to the MD to initiate an AAP session (R10-130).
- Long-press **MUST** trigger AAP ASR regardless of video focus (R10-140)
- AAP ASR **SHOULD** report `KEYCODE_SEARCH` keycode to the MD within 500ms of key-down (R10-150).
- AAP ASR **MUST** report `KEYCODE_SEARCH` keycode to the MD within 600ms of key-down (R10-160).

#### 10.2.3. Cancelling or restarting an ASR session

In addition to starting an ASR session, hardware buttons are also used to indicate the session should be cancelled or restarted. During an active AAP ASR session:

- Short-press of the ASR hardware button MUST cancel the active session (R10-170).
- Long-press of the ASR hardware button MUST restart the session (R10-180).
- KEYCODE\_SEARCH keycode MUST be sent using the **reportKey** function as soon as the ASR hardware button is pressed down (R10-190).

### 10.3. ASR and reverse backup camera

When the backup camera is showing on the AAP projected area, and a user initiates an ASR (either through hotword detection or the ASR hardware button), the HU MUST grant any audio focus requests or microphone open requests and MUST reject any video focus requests sent by the MD (R10-191). The HU MAY ignore presses on the ASR hardware button when the backup camera is showing (R10-192).

If the user puts the vehicle in reverse during an ongoing ASR session, the HU MUST NOT end the ASR session (R10-193).

### 10.4. Speech audio format

To ensure proper functionality, the microphone and HU MUST support the audio format listed in the following table (16-bit Mono PCM, 16KHz sampling rate and buffer size of 2048) (R10-200). The HU MUST open the microphone and start recording within 500ms of receiving a microphone open request (R10-201).

Format	PCM
Sampling Rate	16KHz
Sample Size	16Bits
Buffer Size	2048 (frames MUST be multiples of Buffer Size)
File Type	.wav
Stereo Type	Mono

#### 10.4.1. Processing requirements

Noise Reduction Processing:

- MUST be disabled for single-mic systems (R10-210)
- MAY be enabled for multiple-mic systems that use techniques such as beamforming to improve results in adverse conditions (R10-220)

Automatic Gain Control (AGC):

- MUST be disabled (R10-230)

AAP ASR uses its own AGC to adjust for audio volume. Interactions between third-party and Google AGC are generally poor, leading to reduced overall quality.

## 10.4. Testing recommendations

- Device SHOULD exhibit flat amplitude versus frequency characteristics (+/- 3 dB 150 Hz to 7000 Hz) (R10-240).
- Audio input sensitivity MUST be set such that 90 dB SPL at 1000 Hz yields RMS of 2500 for 16-bit samples (R10-250).
- Harmonic distortion SHOULD be less than 1% (total) from 100 Hz to 4000 Hz at 90 dB SPL input level (R10-260).
- A high pass filter SHOULD attenuate sensitivity to frequencies below 20 Hz by at least 30 dB (R10-270).

## 10.5. Branding ASR

The term **Google Assistant** MAY be used on native on screen displays or messages to indicate voice interactions with the Google App (R10-280). For example, for a user message that explains how to access Google Assistant in Android Auto, the correct usage is "Long press the Push To Talk button to talk to your Google Assistant". Incorrect branding of Android Auto's ASR features such as "Google Voice", "Google Voice Search", "Google Search", "Google voice actions", and "Android Auto voice actions" are not compliant with Google branding and MUST NOT be used (R10-285).

## 10.6. ASR session notifications

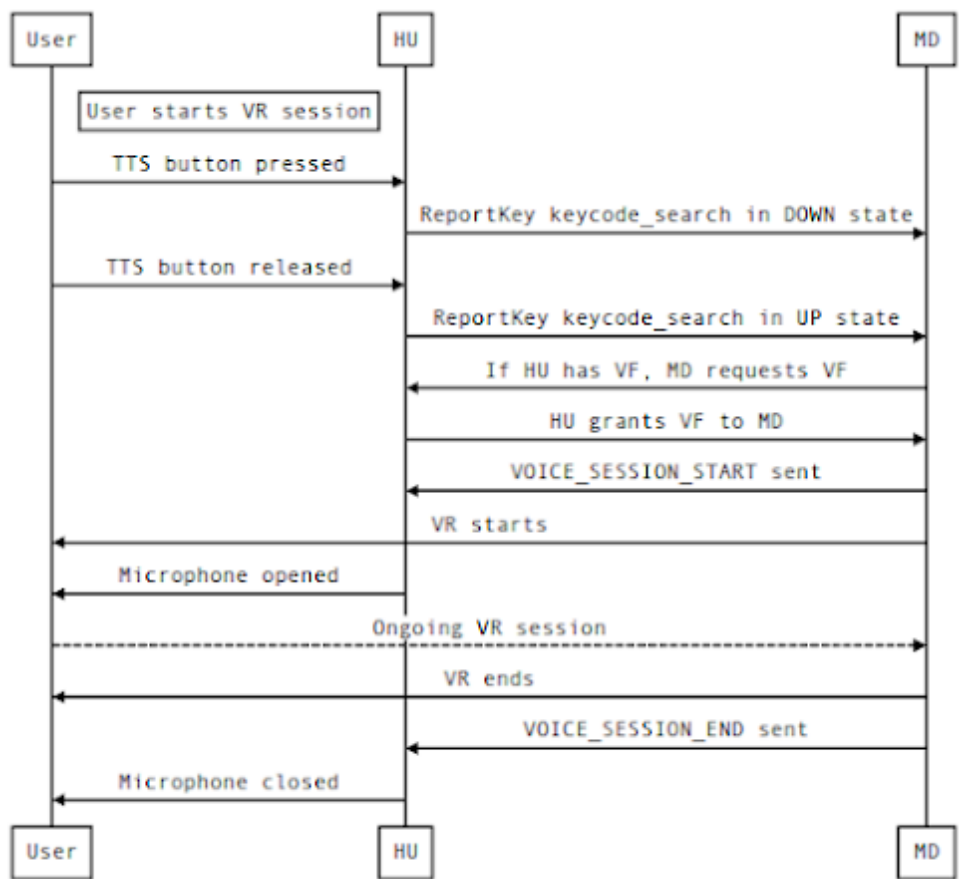
AAP ASR occurs within a Voice Recognition (VR) session, which is bounded by matching `voiceSessionNotificationCallbacks` from the MD:

- An asynchronous `voiceSessionNotificationCallback` is sent on the registered controller callback when the VR session starts.
- A second `voiceSessionNotificationCallback` is sent when the VR session is completed.

## 10.7. ASR short-press implementation

The following flow diagrams map a short-press ASR implementation in detail.

### Short Press Standard Session



## Short-Press Restart

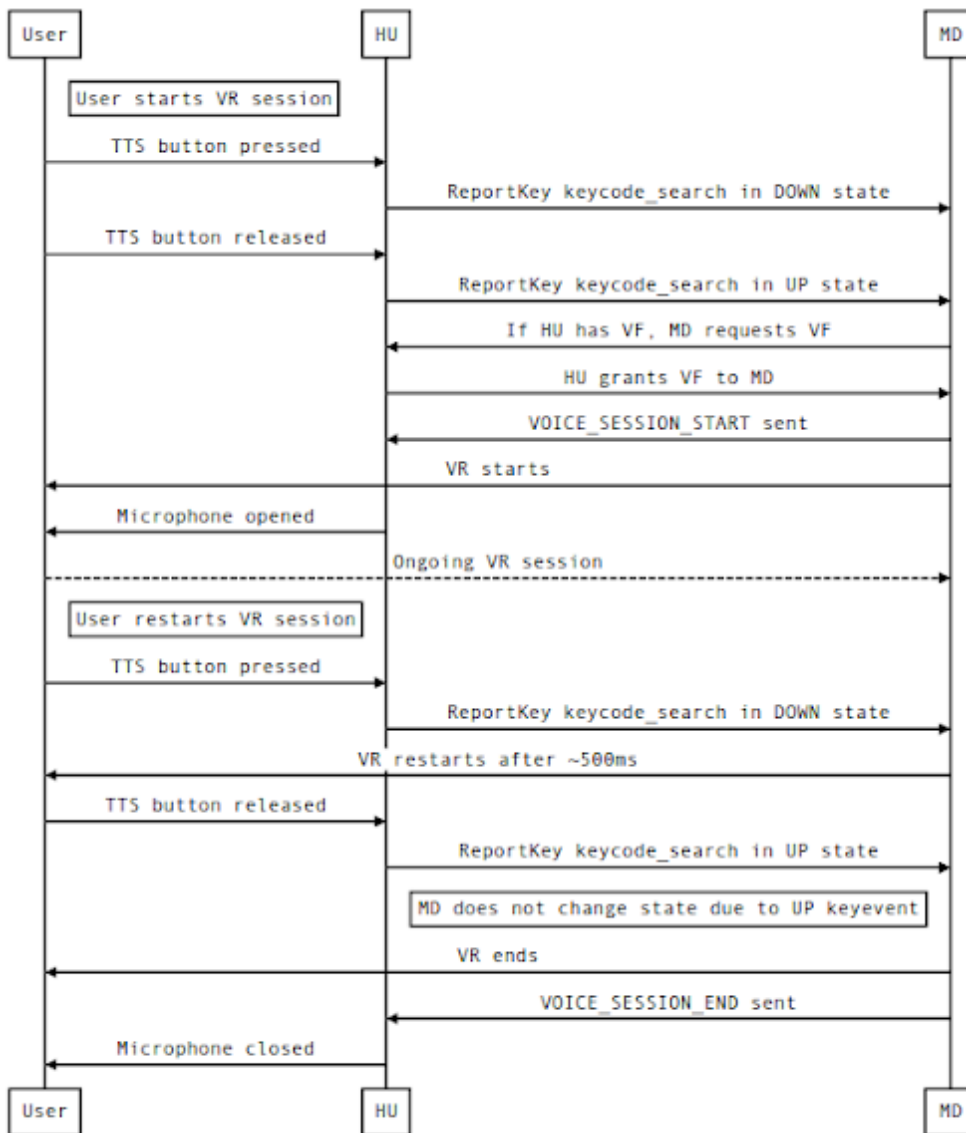
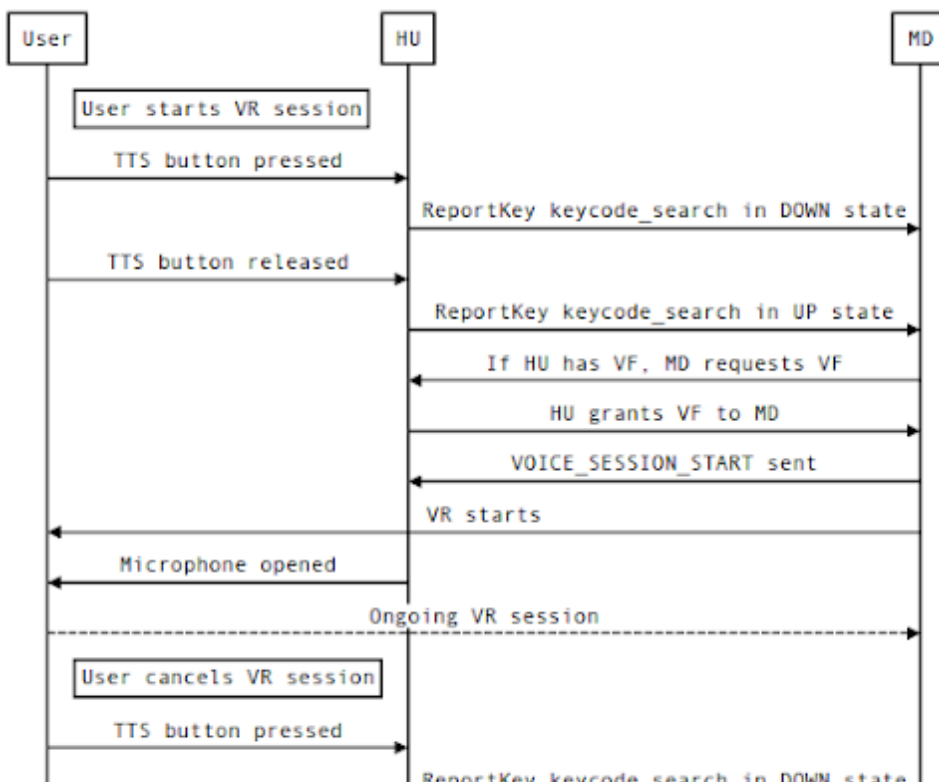


Figure 10.2. Short-press, standard implementation, restart.

## Short-Press Cancel



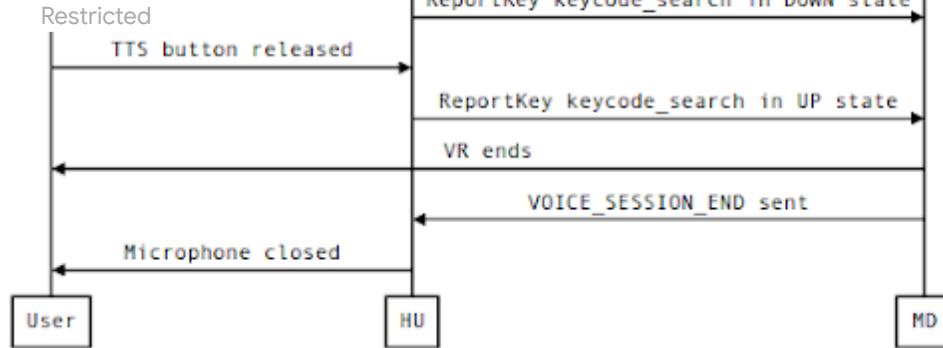


Figure 10.3. Short-press, standard implementation, cancel.

## 10.8. ASR long-press implementation

The following flow diagrams map a long-press ASR implementation in detail.

### Long-Press Standard Session

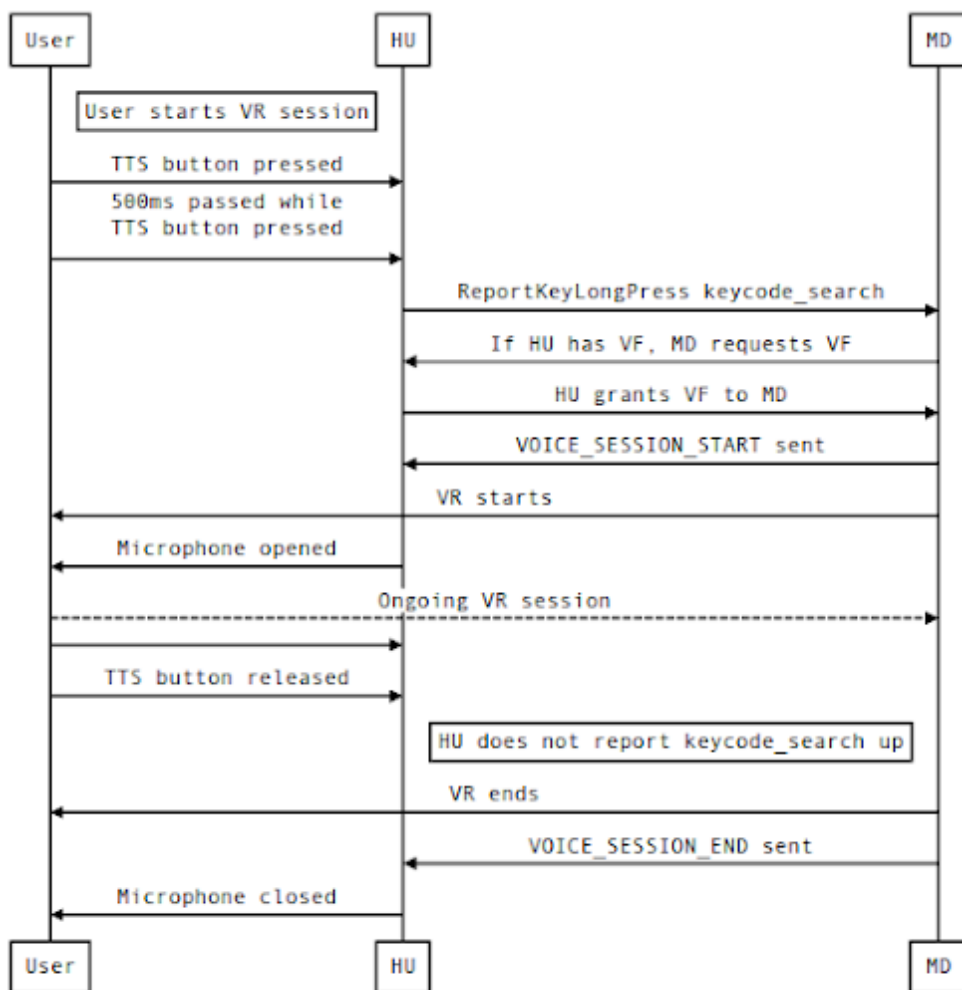
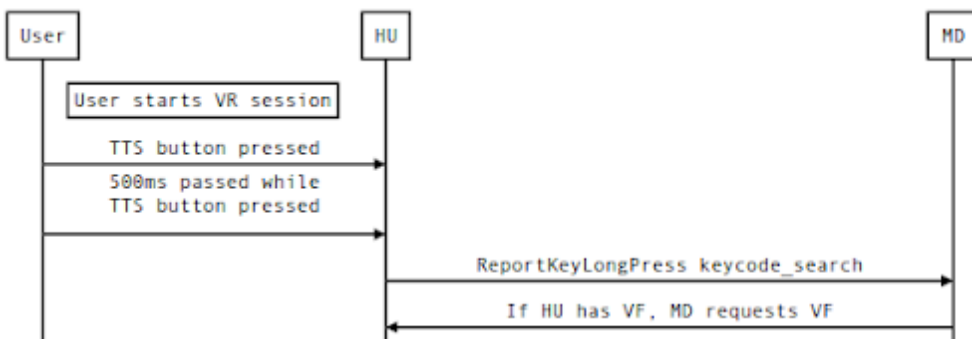


Figure 10.4. Long-press, standard session.

### Long-Press Restart



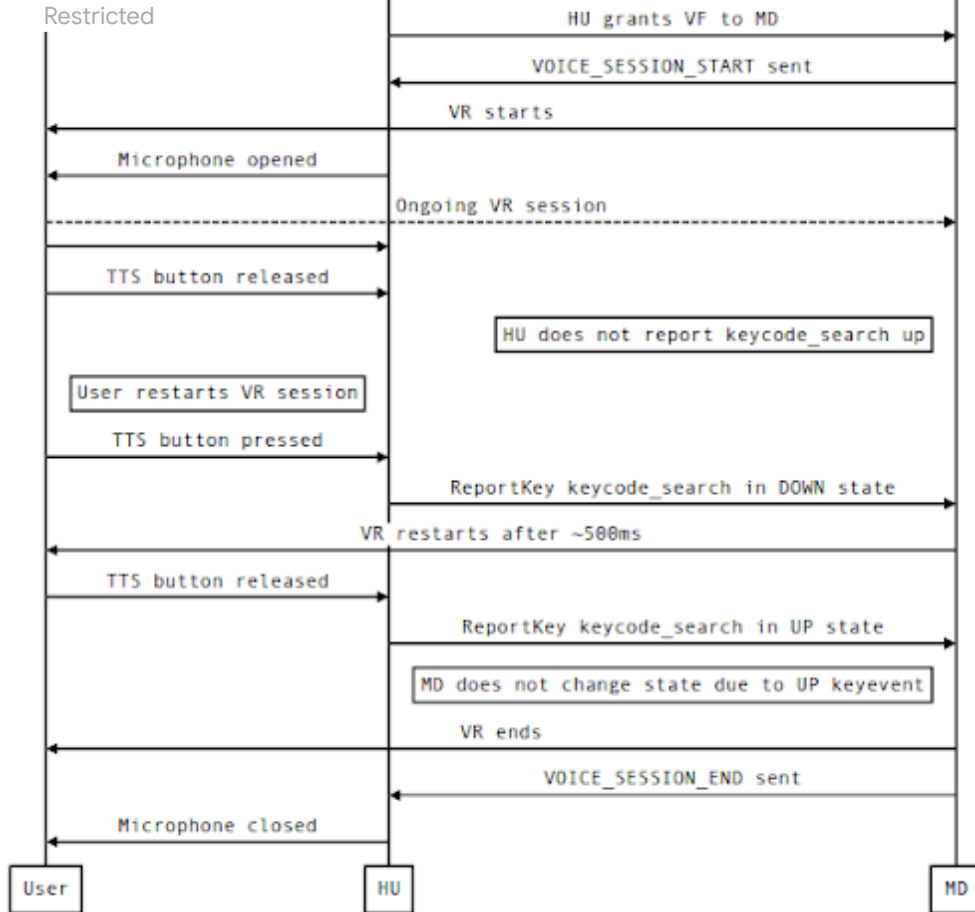
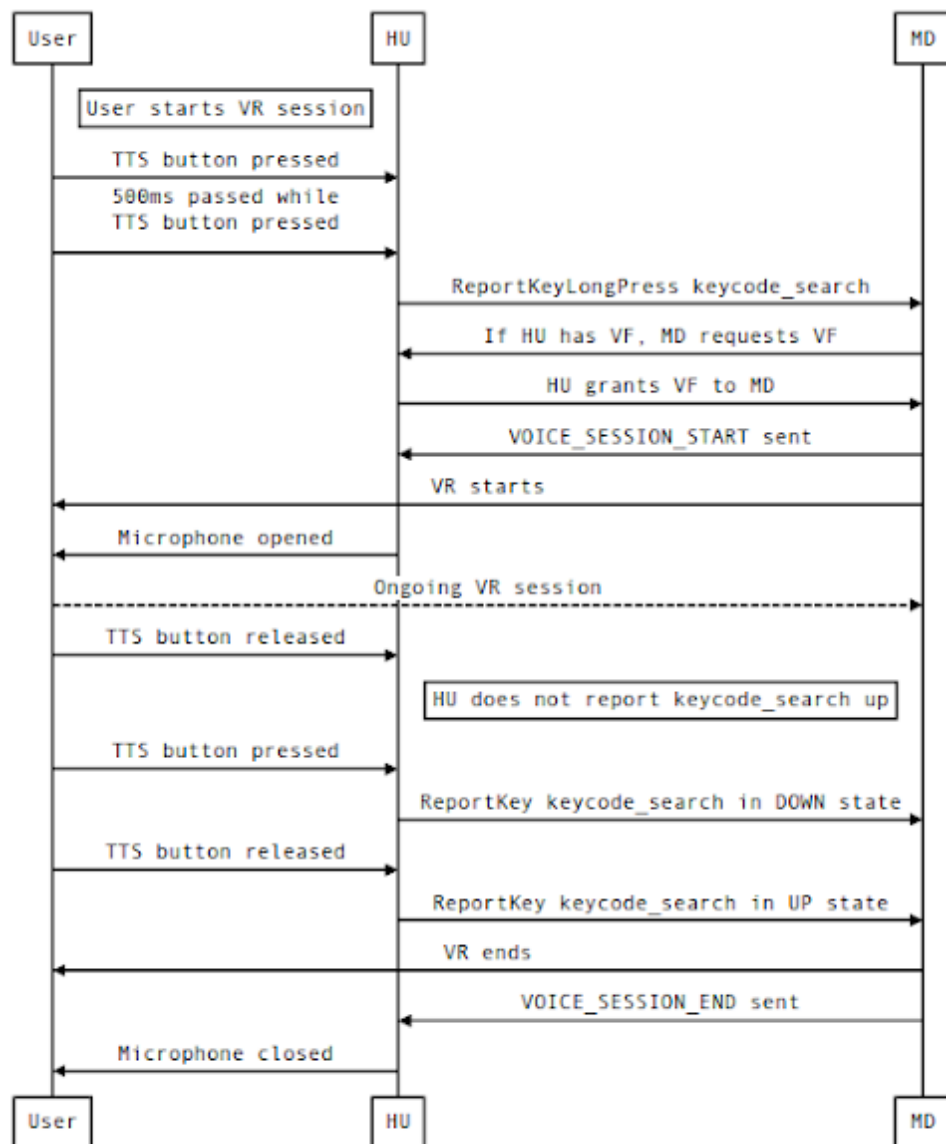


Figure 10.5. Long-press, restart.

## Long-Press Cancel





**Figure 10.6.** Long-press, cancel.

## 11. Navigation

Because the MD provides turn-by-turn navigation and some HUs also provide native (on-board) navigation, the MD and HU **MUST** negotiate navigation focus so the user does not receive concurrent navigation and traffic instructions (R11-010).

AAP includes two navigation focus (NF) modes: **Projected** (navigation provided from MD) and **Native** (navigation provided from HU). On AAP connection, the default NF mode is **Native**. The HU controls navigation focus and determines which navigation service runs.

The HU **MUST** implement navigation focus even if it does not include native navigation (R11-020). The HU **MUST** handle navigation focus independently of audio focus and video focus (R11-030).

The intent of navigation focus is to ensure that concurrent navigation from two different sources does not happen. The intent is not to block users from accessing native navigation features while an AAP session is active. Accordingly an HU that includes native navigation:

- **MUST** allow native maps to be viewed when an AAP session is active, regardless of navigation focus mode (R11-031)
- **MUST** allow native navigation to be started when an AAP session is active (R11-032)

### 11.1. Navigation focus request

A navigation focus request is a message sent from the MD to the HU. The MD makes a request for navigation focus whenever the user changes the navigation state in the MD:

- When navigation on the MD is started or restarted, the MD makes a navigation focus request to the HU, requesting NF mode to be Projected.
- When navigation on the MD is stopped, the MD makes a navigation focus request to the HU, requesting NF mode to be Native.

The HU **MUST** grant all navigation focus requests from the MD (R11-040). When the HU grants a request from the MD for Projected navigation focus, it **MAY** display a message to the user indicating that Native navigation has stopped (R11-050).

### 11.2. Navigation focus notification

A Navigation focus notification is a message sent from the HU to the MD that specifies whether the NF mode is Projected or Native and is used to grant navigation focus. The HU:

- **MUST** send a navigation focus notification when it starts Native navigation and **MAY** send a navigation focus notification when it stops Native navigation (R11-060)
- **MUST** send a navigation focus notification to the MD in response to a navigation focus request from the device (even if there is no change in navigation focus) (R11-070)

The MD will not start phone navigation until it receives a navigation focus notification with navigation focus set to Projected. The HU **MUST** only run native navigation when the navigation focus is set to Native (R11-080).

### 11.3. Integrating navigation

AAP includes two NF modes: **Projected** (navigation provided from MD) and **Native** (navigation provided from HU). Each time an AAP connection starts, the default NF mode is **Native**. AAP enables users to switch navigation focus between Projected and Native. However, the **HU always controls navigation focus** and determines which navigation service runs.

## 12. Vehicle data

When connected to a AAP MD, the vehicle HU sends read-only vehicle information to the MD using a pub-sub model. The MD registers for specific events and indicates an update frequency.

Vehicle data:

- MUST always be made available, regardless of the current video and audio focus state (R12-010)
- MUST NOT be faked, simulated, or falsified (R12-020)
- MUST be notified to the MD upon initial request for an event (if data is available) (R12-030)
- MUST be sent to the MD upon request from the MD (R12-031)
- MUST NOT be sent to the MD without a request from the MD (R12-032)

The HU MUST NOT report a vehicle data sensor as available during Service Discovery if the HU cannot (or will not) send data from that sensor to the MD (R12-033). Sensor errors MAY be reported to MD using the `SensorError` message when identified by the HU (R12-040). The HU MAY use the 'updateServiceDiscovery' message to update those services for which support has already been declared. After service discovery is complete, this message can be sent unsolicited at any time.

### 12.1. Navigation data

#### 12.1.1. GNSS

To ensure good performance during navigation, an OEM HU MUST have GNSS (R12-045).

If the HU has GNSS, it MUST send GNSS location data to the MD (R12-060). If the HU does not have GNSS, the vehicle MUST specify a placement location for the MD that allows the MD to achieve optimal GNSS reception considering the constraints of the placement in a vehicle, e.g. indicate a location at the top of the dashboard (R12-070). If the HU does not have GNSS, the MD GNSS is used instead.

The HU sends GNSS location data and GNSS satellite status to the MD independently using the `reportLocationData` and `reportGpsSatelliteData` methods respectively. For details on parameters, refer to the [Receiver Library source code](#).

HUs with GNSS MUST support the `LocationCharacterization` field in the `SensorSourceService` declaration (R12-120).

GNSS location data:

- MUST be sent at the same update rate as the underlying hardware (R12-080)
- MUST support location outputs at a rate of at least 1 Hz (R12-085)
- MAY be dead reckoned (DR) values by fusing either the calculated GNSS position or GNSS satellite measurements with additional sensors (R12-100)
  - If the location data is dead reckoned, data from all sensors used MUST be provided to the MD (R12-105)
- MUST NOT be map matched values (R12-110)
- MUST provide data for all the fields in `LocationData` (incl. expected horizontal position accuracy (68% radial confidence), altitude, speed and bearing) (R12-125)
- MAY be sent to MD when no GNSS position fix is available (R12-130)

GNSS satellite status:

- MUST be sent at the same update rate as the underlying hardware (R12-140).
- MUST accurately report the number of GNSS satellites used in determining the location (R12-150).

- MUST have the number of GNSS satellites used in determining the location set to zero (0) when no satellites are being used (R12-160).
- MUST have the number of GNSS satellites used in determining the location set to greater than zero (0) otherwise (R12-170).
- MUST be sent all the time, regardless of whether location fix was acquired or not (R12-180).
- SHOULD include the number of GNSS satellites in view (R12-190)
- SHOULD report GNSS satellite status as soon as satellites are tracked, even if a location calculated from GNSS is not yet reported (R12-203).
- SHOULD report complete GNSS satellite status (i.e. all fields of **GpsSatellite** are valid for each satellite reported) from all constellations tracked (as reported in **GpsSatelliteData**) with the exception of SBAS (R12-205).
  - The current protocol version only supports GPS satellites, support for other constellations will be added in future revisions.

For GNSS performance, the HU:

- SHOULD be able to determine the location in open-sky conditions (strong signals, negligible multipath, HDOP < 2) within 10 seconds (fast time to first fix), when connected to a 0.5 Mbps or faster data speed internet connection. This is typically achieved by the use of Assisted or Predicted GNSS technique to minimize GNSS lock-on time (Assistance data includes Reference Time, Reference Location and Satellite Ephemeris/Clock).
- MUST determine its location in open sky within 5 seconds, when location requests are restarted, up to an hour after the initial location calculation, even when the subsequent request is made without a data connection and/or after a power cycle (R12-207).
- MUST be able to determine location within 20 meters and speed within 0.5 meters per second at least 95% of the time in open sky conditions after determining the location (R12-208).
- MUST simultaneously track at least 8 satellites from one constellation (R12-209).
- SHOULD be able to simultaneously track at least 24 satellites, from multiple constellations (e.g. GPS + at least one of Glonass, Beidou, Galileo).

### 12.1.2. Other navigation sensors

If the HU has any other navigation sensors, data from those sensors MUST be provided over AAP (R12-210). OEM HUs MUST provide mechanical speed (R12-220) and SHOULD provide 3-axis accelerometer data (R12-223). Aftermarket HUs that support only USB projection SHOULD provide mechanical speed, 3-axis accelerometer data and 3-axis gyroscope data (R12-230).

Mechanical speed:

- MUST be sent at the same update rate as the underlying hardware, with a desired rate of 10 Hz, minimum of 1 Hz and maximum of 30 Hz (R12-240)
- MUST be derived from mechanical rotation at transmission or wheels (not GPS) (R12-250)
- MUST be a negative value when vehicle is moving in reverse (R12-260)

Compass heading:

- MUST be sent at the same update rate as the underlying hardware, with a desired rate of 10 Hz, minimum of 1 Hz and a maximum of 20 Hz (R12-270).
- MAY be 1-axis compass heading or 3-axis if pitch and/or roll are available (R12-280).
- MUST indicate the direction of the front of the vehicle (R12-300).
- MUST NOT be GPS bearing data: data MUST NOT be sent if the HU does not have access to a compass heading

Restricted  
Compass heading data MUST NOT be sent if the HU does not have access to a compass heading that is independent of the location course) (R12-310).

- SHOULD send values showing magnetic North (rather than true North) (R12-320).
- SHOULD have a resolution of 1 degree (R12-330).
- MUST have a resolution of less than 45 degrees (R12-340).
- MAY be gyro stabilized (R12-350).

Accelerometer:

- MUST be sent at the same update rate as the underlying hardware, with a desired rate of 10 Hz, minimum of 3 Hz and a maximum of 20 Hz (R12-360).
- MUST be sent so that positive X axis points to the right door, positive Y axis point to the nose of the car, and positive Z axis points to the roof of the car (R12-370); for details refer to [Automotive axes](#).

Gyroscope:

- MUST be sent at the same update rate as the underlying hardware, with a desired rate of 10 Hz, minimum of 3 Hz and a maximum of 20 Hz (R12-380).
- MUST include all 3-axis gyroscope data (R12-390).

## 12.2. Other Vehicle data

### 12.2.1. Required data

The HU MUST send Driving Status and Day/Night Mode (R12-410).

#### 12.2.1.1. Driving Status

Driving Status indicates the level of feature/functionality lockout as determined by vehicle and is defined as a bitmask.

Driving Status:

- MUST be selected in compliance with the laws and regulations that apply to markets where the product is shipping (R12-420)
- MUST have the "No setup/configuration allowed" bit set to 0 when the vehicle is parked (R12-430)
- MUST have the "No setup/configuration allowed" bit set to 1 when the vehicle is not parked (R12-431)
- SHOULD have the "No text input allowed" bit set to 0 when the vehicle is parked (R12-440)
- SHOULD have the "No text input allowed" bit set to 1 when the vehicle is not parked (R12-441)

#### 12.2.1.2. Day/Night Mode

Day/Night Mode indicates the use of night mode when rendering the UI. Day/Night Mode:

- MUST be consistent with dashboard day/night mode (R12-460)
- MUST be either set by the user or set automatically by the HU (R12-470)
- SHOULD provide the user with an option to manually override the HU setting if set automatically by the HU (R12-480)
- SHOULD have signal de-bounced to avoid flickering if using ambient light sensor input to set automatically (R12-490)

### 12.2.2. Fuel Type and EV Connector Type

Fuel Type indicates which type of fuel a vehicle uses. Fuel Type:

Restricted

- SHOULD be provided by the HU (R12-491)
- if sent, MUST include all fuel types the vehicle supports (R12-492)

EV Connector Type indicates which type of EV charging connection a vehicle uses. EV Connector Type:

- MUST be sent if an HU sends a Fuel Type of **FUEL\_TYPE\_ELECTRIC** (R12-493).
- MUST be sent in order of preference, for example prefer fast charge (R12-494).

### 12.2.3. Optional data

The following vehicle data items are optional and MAY be sent to the MD (R12-500). Vehicle data should only be sent if actual data can be sent; synthesized values MUST NOT be sent (R12-510). For details on optional data items, refer to the [Receiver Library source code](#).

Data	Rate (Desired/Min/Max)
<i>RPM</i>	10 / 1 / 30 Hz
<i>Current Gear</i>	on change
<i>Parking Brake</i>	on change
<i>Odometer</i>	on change
<i>Estimated Range</i>	on change
<i>Fuel Level Low</i>	on change
<i>Passenger Presence</i>	on change
<i>Door/Trunk/Hood status</i>	on change
<i>Headlights</i>	on change
<i>Tire Pressure</i>	on change
<i>Environmental Data</i>	1.0 / 0.1 / 2.0 Hz
<i>HVAC Data</i>	1.0 / 0.0 / 2.0 Hz
<i>Toll Card</i>	on change

Methods **reportDiagnosticsData** and **reportDeadReckoningData** in the Receiver Library are deprecated and MUST NOT be used in new integrations (R12-520).

12.2.3.1. ETC Icon Display

Toll roads in Japan accept payment via an Electronic Toll Collection (ETC) system that utilizes a removable card. If drivers without an ETC reader or without a valid ETC card inserted in the reader attempt to go through a toll gate, the bars of the toll gate automatically shut. To ensure drivers can easily check whether they have their ETC card inserted and avoid toll gate accidents, many native navigation systems in Japan include an "ETC" icon in their UI. From Receiver Library v1.5, Android Auto also provides a mechanism to enable this functionality, with an ETC Icon displayed inside the AAP UI.

An HU MAY report a SENSOR\_TOLL\_CARD sensor during Service Discovery (R12-600). If SENSOR\_TOLL\_CARD sensor is reported, the HU MUST report the current state of an ETC card (true: ETC card is present and valid, false: other cases) when the state is changed (R12-610).

SENSOR\_TOLL\_CARD sensor MUST only be reported in HUs shipping in Japan (R12-620).

12.3. Driving Status bitmask definition

The Driving Status bitmask is defined in the table below.

00000	Unrestricted.
00001	No video playback (movies, YouTube, games, etc.) allowed. Does not affect UI.
00010	No text input allowed (on-screen keyboard, rotary controller speller, touchpad text entry). Limits UI interaction.
00100	No voice input allowed.
01000	No setup/configuration allowed.
10000	Limit displayed message length.

13. Application status and data

The primary visual UI for AAP renders on the MD and projects to the in-vehicle HU display as a projection video stream. To enable tighter integration with native purpose-built UI such as instrument clusters, heads-up displays and native widgets on the HU home screen, AAP also provides limited access to app data.

13.1. Navigation data

Vehicles that display navigation data in their instrument cluster, heads-up display, or on any other device from their native navigation solution MUST also display navigation data from AAP (R13-001). The HU subscribes to next turn updates from the MD by implementing [NavigationStatusEndpoint](#) and [INavigationStatusCallbacks](#).

If the vehicle displays navigation data from AAP, the vehicle:

- MUST include a turn indicator icon to represent the next maneuver (R13-002)
- MUST include distance to the next maneuver (R13-003)

• MUST include street name (R13-004)

- MUST ensure icons and text match (as closely as possible) the AAP navigation guidance (R13-005)

If the navigation widget is selectable, selecting the widget MUST send KEYCODE\_NAVIGATION to the MD to show the AAP Maps facet (R13-006).

### 13.1.1. Navigation subscription

To receive navigation data, the HU sends a navigation subscription request to the MD during service discovery. Options for this request are detailed in the table below.

Subscription Field	Description
<i>Minimum Interval</i>	Minimum interval between Next Turn Distance events (in ms).
<i>Next Turn Format</i>	<ul style="list-style-type: none"><li>• <b>Image.</b> High-fidelity next turn iconography rendered on the MD.</li><li>• <b>Enum.</b> HU selects and renders a native next turn icon based on the turn event enum sent by the MD.</li></ul>
<i>Image Options</i>	Required only for HUs that support next turn images. Includes color depth (8, 16, or 32 bits) and resolution (up to 512 x 512 pixels). Square images only.

Instrument clusters that can dynamically load images MUST select IMAGE as the Next Turn Format (R13-010). The next turn image may have more variations than expressed by the next turn enum value. For example, variations in the sharpness of ON\_RAMP turns may be expressed with different images but not be distinguished by this enum.

### 13.1.2. Navigation status event

The MD sends navigation status event messages to the HU.

Event Field	Description
<i>Navigation Status</i>	Current status of mobile-based navigation (ACTIVE, INACTIVE, UNAVAILABLE).

### 13.1.3. Next turn event

The MD sends next turn event messages to the HU.

Event Field	Description
<i>Next Turn Road Name</i>	Name of next road (UTF-8).
<i>Next Turn Side</i>	Enumeration describing the side of turn (left, right, unspecified)
<i>Next Turn Enum</i>	Enumeration describing the type of turn.
<i>Next Turn Image</i>	PNG formatted turn image.



### 13.1.4. Next turn distance event

The MD sends next turn distance event messages to the HU. These events are sent continuously from MD to HU while navigating, and not necessarily at predictable distance milestones.

Event Field	Description
<i>Next Turn Distance</i>	Distance (meters) to the next turn.
<i>Next Turn Seconds</i>	Estimated time (seconds) to the next turn.
<i>Next Turn Rounded Distance</i>	Distance to the next turn, rounded and converted to the units used when this distance is displayed by the MD navigation application. This value is scaled by 1000. For example, when the distance is 1.5 km, this value will be set to 1500 and the unit set to kilometer.
<i>Rounded Distance Units</i>	The units used on the <i>Next Turn Rounded Distance</i> .

The MD sends distance values in meters and may not time distance events to occur at round number intervals (2km, 1km, 500m, 100m, and so on). The HU MUST use Next Turn Rounded Distance to display a distance value and units consistent with the MD navigation application (R13-020).

### 13.1.5. Next turn enum

The MD sends next turn enum messages to the HU.

Enum Value	Has Side	Has Count	Icon	Description
DEPART	N	N		Start driving. Used before accurate location or orientation is known.
NAME_CHANGE	N	N		Indicate a street name change without turn.
SLIGHT_TURN	Y	N		Make a 10 – 45 degree turn. Shown for <b>side=RIGHT</b> ; reflected for <b>side=LEFT</b>
TURN	Y	N		Make a 45 – 135 degree turn. Shown for <b>side=RIGHT</b> ; reflected for <b>side=LEFT</b>
SHARP_TURN	Y	N		Make a 135 – 175 degree turn. Shown for <b>side=RIGHT</b> ; reflected for <b>side=LEFT</b>
U_TURN	Y	N		Make a 180 degree turn onto the same road. Shown for <b>side=RIGHT</b> ; reflected for <b>side=LEFT</b>
Enum Value	Has Side	Has Count	Icon	Description

Restricted

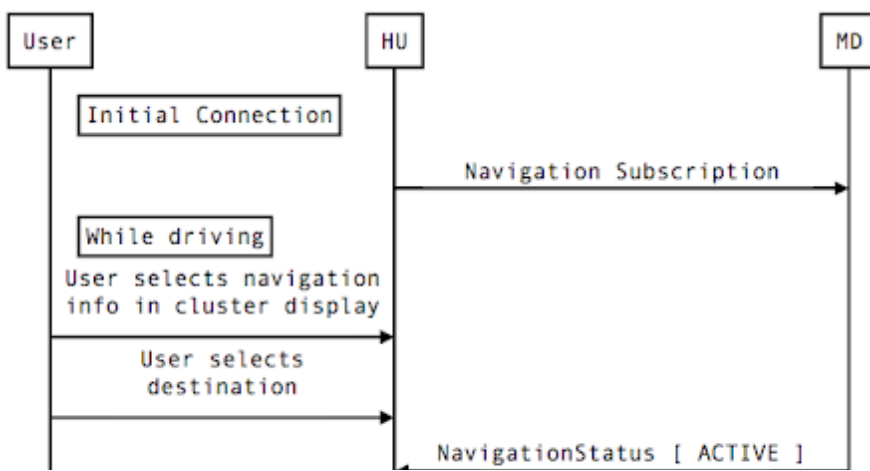
ON_RAMP	Y	N		Take an on-ramp onto a major road. Shown for <b>side=RIGHT</b> ; reflected for <b>side=LEFT</b>
OFF_RAMP	Y	N		Take an off-ramp from a major road. Shown for <b>side=RIGHT</b> ; reflected for <b>side=LEFT</b>
FORK	Y	N		Take a fork while on a ramp. Shown for <b>side=RIGHT</b> ; reflected for <b>side=LEFT</b>
MERGE	Y	N		Merge with another road.
ROUNDABOUT_ENTER	Y	N		Enter a large roundabout. Shown for <b>side=RIGHT</b> ; reflected for <b>side=LEFT</b>
ROUNDABOUT_EXIT	Y	N		Exit a large roundabout. Shown for <b>side=RIGHT</b> ; reflected for <b>side=LEFT</b>
ROUNDABOUT_ENTER_AND_EXIT	Y	* Y/N		Enter a roundabout then take the nth exit.
				ROUNDABOUT enums are described in the table below.
				Shown for <b>side=RIGHT</b> ; reflected for <b>side=LEFT</b> .
				* The <b>COUNT</b> code> field may not be set.
				
				
				
				
				
STRAIGHT	N	N		Indicates a potentially confusing intersection, in which the user is to proceed straight.
FERRY_BOAT	Y	N		Board a boat ferry.
Enum Value	Has Side	Has Count		Description

FERRY_TRAIN	Y	N		Board a train ferry (Google Maps displays no icon).
DESTINATION	Y	N		Indicated arrival at destination. Shown for <b>side=RIGHT</b> ; reflected for <b>side=LEFT</b>

The **ROUNDAABOUT\_ENTER\_AND\_EXIT** (**value=13** in the table above) icons are described below, according to turn angle.

Turn Angle (In Degrees)	Icon	Description
1 – 67		Shown for <b>side=RIGHT</b> ; reflected for <b>side=LEFT</b>
68 – 112		Shown for <b>side=RIGHT</b> ; reflected for <b>side=LEFT</b>
113 – 157		Shown for <b>side=RIGHT</b> ; reflected for <b>side=LEFT</b>
158 – 202		Shown for <b>side=RIGHT</b> ; reflected for <b>side=LEFT</b>
203 – 247		Shown for <b>side=RIGHT</b> ; reflected for <b>side=LEFT</b>
248 – 292		Shown for <b>side=RIGHT</b> ; reflected for <b>side=LEFT</b>
293 – 337		Shown for <b>side=RIGHT</b> ; reflected for <b>side=LEFT</b>
338 – 360		Shown for <b>side=RIGHT</b> ; reflected for <b>side=LEFT</b>

### 13.1.6 Navigation data example



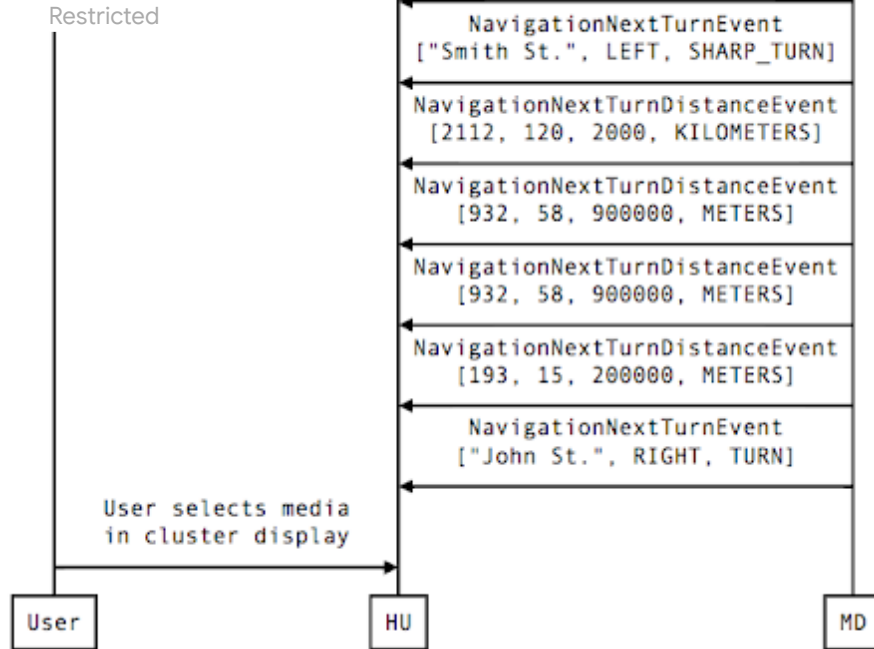


Figure 13.1. Navigation data example.

## 13.2. Accessing media data

Vehicles that display native media playback status and/or metadata in their instrument cluster, etc. MUST also display AAP media playback status and/or metadata (R13-030). The HU subscribes to media playback and metadata by implementing `MediaPlaybackStatusEndpoint` and `IMediaPlaybackStatusCallbacks`.

If the vehicle's instrument cluster or other secondary screen (i.e heads-up display) is capable of displaying a source-specific graphic or icon, the Android Auto Label Icon MUST be displayed (R13-031). If graphics or icons are not supported, "Android Auto" as text MUST be used (R13-032). If a media source is displayed, the source specified in the Media Playback Status message MUST be displayed (R13-033). The cluster or HU's media widget MUST NOT use "USB" or display the USB logo when displaying AAP media data (R13-034). When the HU receives media data updates, the HU MUST refresh the display data in the cluster (R13-035).

The Receiver Library defines a `MediaBrowserEndpoint` that is intended to allow browsing of media sources over the GAL protocol. However, `MediaBrowserEndpoint` is not supported on the MD side and updates will not be sent. Accordingly, HU MUST NOT implement `MediaBrowserEndpoint` integration (R13-040).

### 13.2.1. Media playback status

If the HU subscribes to media data, the MD continually sends status and metadata notifications whenever media is playing. Status notifications include the following data:

Field	Description
<i>State</i>	PLAYING, PAUSED, STOPPED
<i>Source</i>	The name of the audio application source. For example, "Google Play Music", "Pandora", "TuneIn Radio", etc.
<i>Playback</i>	Current progress in track in seconds.
<i>Shuffle</i>	TRUE if shuffle is selected.
<b>Field</b>	<b>Description</b>
<i>Repeat</i>	TRUE if repeat is selected.

Restricted	
Repeat One	TRUE if repeat-one-track is selected.

### 13.2.2. Media playback metadata

If the HU subscribes to media data, the MD continually sends status and metadata notifications whenever media is playing. Metadata notifications include the following data:

Field	Description
Song	Name of the current song/track.
Artist	Name of the current artist.
Album	Name of the current album, if available.
Album Art	PNG image of the album art for the currently playing track, if available. Resolution can be up to 256px x 256px.
Playlist	Name of the current playlist, if available.
Duration	Duration of this track in seconds.
Rating	Rating on a 0-5 scale, if available.

### 13.3. Accessing Telephony data

Vehicles that display telephony data in their instrument cluster, heads-up display, etc. for MDs connected over HFP-only MUST display telephony data for phones connected over AAP (R13-050).

## 14. Vendor Extension Channels (VECs)

The Android Auto Protocol (AAP) supports protocol extensions in vendor-specific channels that use AAP as a transport. These Vendor Extension Channels (VEC) allow OEMs to pass proprietary data between the HU and MD. For example the VEC could be used to:

- Pass data on engine performance to an Android Auto app distributed by the OEM to display to users on the HU screen in Projection.
- Collect service data from the vehicle and send to a background service running on the MD without a UI, and upload to the cloud.

For information on how to develop Android Auto full screen applications that can display information received from the vehicle via a VEC, refer to the [Android Auto SDK documentation](#).

### 14.1. VEC architecture



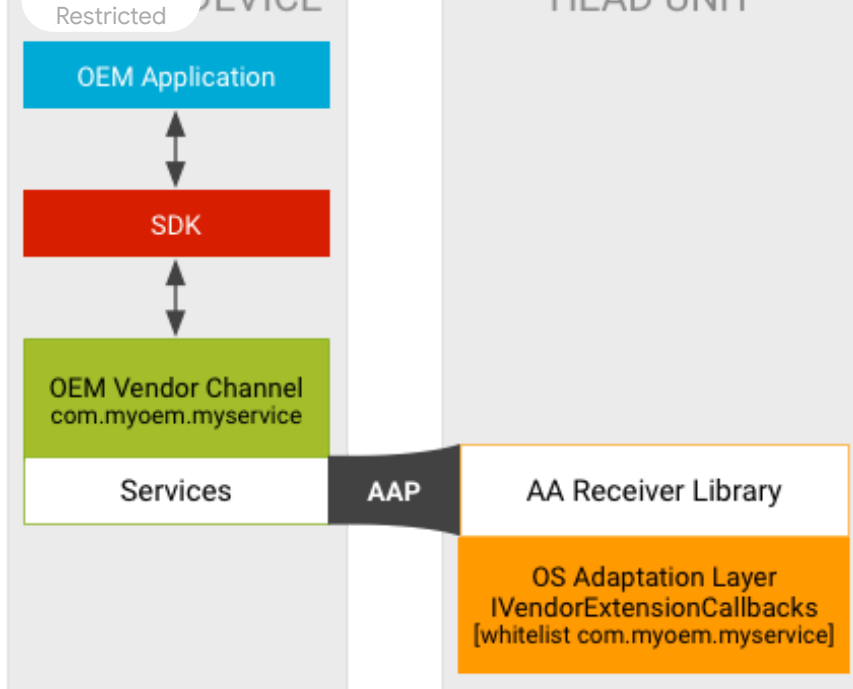


Figure 14.1. VEC architecture.

The HU MAY implement multiple VECs, and multiple VECs can be active at the same time (R14-010).

### 14.1.1. VEC setup flow

The HU MAY send data at any time, regardless of whether the MD is ready to receive it (R14-020). The setup flow for the MD to be ready to communicate over the VEC is as follows:

1. After starting, a Service within an OEM App creates a Car instance using `android.support.car.Car.createCar()` and then registers a `CarConnectionCallback` listener to receive car connection notifications.
2. When the Service receives a connection notification, it loads the `CarVendorExtensionManager`.
3. The Service registers a `CarVendorExtensionListener` on the `CarVendorExtensionManager` which calls `onData()` in the listener when data arrives.
4. The Service can also send data using the `sendData()` call.

### 14.1.2. VEC security

All AAP traffic is sent over a connection encrypted with SSL, including data sent over the VEC. Data received over the AAP connection by the MD has the SSL encryption removed by `CarService` (part of Google Play Services) before passed on the OEM App. OEMs SHOULD add further encryption to data they send over the VEC (R14-030).

The HU MUST specify the package name of an Android service or app that is allowed to connect to the VEC during service discovery (R14-040). This is set in the HU in `VendorExtension::addDiscoveryInfo()`. For details, see `add_package_white_list()` call in `addDiscoveryInfo` in the [VEC sample HU implementation](#).

On the MD, when an OEM app is started and requests a VEC connection, Android Auto checks that the service trying to connect to the VEC passes the standard Android Auto permissions checks and has a package name that matches the one declared by the HU vendor extension implementation. If these criteria are satisfied, the nominated OEM service can begin sending and receiving data over the VEC.

**NOTE:** Standard Android Auto permission checks are handled by the Google Play Store validation procedure (not blacklisted, etc.). Whitelisting is required for testing apps not distributed via the Google Play Store (during development/testing). For details, refer to the [Android Auto SDK](#).

### 14.1.3. Using verification certificates

With `Restricted`, Android service is distributing data to multiple apps on the MD, the service SHOULD verify the signing signature of each app that requests VEC access (R14-050). VEC access SHOULD be granted only to apps that are signed with the OEM certificate, thus preventing access by unauthorized app (R14-060).

To limit VEC connections to your own applications, define a `<permission>` with `android:protectionLevel=signature` and protect the VEC service/provider with that permission.

The OEM Service can use `PackageManager.getPackageInfo()` to access a hash of an APK signing signature, then compare this with a known hash of the OEM certificate held within the service.

## 14.2. Implementing the VEC

The payload of each vendor extension message is restricted to 512KB unless that specific extension or message explicitly allows bigger data.

### 14.2.1. MD implementation

An OEM app uses the `CarVendorExtensionManager` to setup its connection to the VEC. The Android Auto [Partner Development Kit \(PDK\)](#) includes a sample EchoVendor app that implements a basic vendor extension example.

OEM apps wanting to use the VEC MUST implement a Service and use that for handling VEC communications (R14-070). Services using the VEC MUST declare the permission `com.google.android.gms.permission.CAR_VENDOR_EXTENSION` in their manifest (R14-080). Only one Service can interact directly with a VEC. Accordingly, if VEC data will be shared by more than one app or service, then one Service MUST handle data processing for the VEC and provide an interface that enables multiple apps to use the data (R14-090).

OEM apps using the VEC MAY provide a UI on the HU by implementing the Android Auto SDK (R14-100). They cannot display a UI on the MD while AAP is connected but can provide a UI for use on the MD after the AAP connection has been terminated.

Users must grant permissions to access car data before an app can connect to the VEC. Usually this requires a UI on the MD or instructions to the user to grant permission in the App Settings menu.

Auto-install of applications is not supported.

#### 14.2.1.1. MD implementation for autostart

The OEM MAY configure their service or app on the MD to start automatically when the MD connects to the HU (R14-110). Apps or services that wish to autostart a VEC connection MUST implement `BroadcastReceiver` and listen for the `com.google.android.gms.car.VENDOR_CHANNEL_READY` broadcast (R14-120).

After AAP initialization, CarService sends a broadcast announcing `VENDOR_CHANNEL_READY` to each package on the VEC whitelist. Any app that receives the directed broadcast will be automatically started by Android, even if the app is not yet running. After this app has started, the standard VEC Setup flow is followed (see [VEC setup flow](#)). If an app that is not whitelisted to open the VEC attempts to open the VEC, the `CarVendorExtensionManager` throws a `SecurityException`.

Before an app can autostart, users must have installed the app on the MD and granted permission to access car data through the Google Play Store Install App permissions screen.

#### 14.2.1.2. MD implementation error handling

Retry and disconnect logic SHOULD be part of any app that communicates with a data service such as the VEC, and as such is not provided by the VEC (R14-130). For example, if AAP is not available because of a failure in the HU OS Abstraction Layer, an OEM app may switch to a different communication method (data over Bluetooth, etc.) or simply store data and wait for reconnection.

### 14.2.2. HU implementation

On the HU, vendor channels subclass `VendorExtension` and `IVendorExtensionCallbacks`. When



51 Restricted `VendorExtension`, implement the pure virtual function  
`VendorExtension::addDiscoveryInfo()`.

For details, see the [VEC sample HU implementation](#) and refer to the following resources:

- `android-protocol-lib/jni/com_google_android_projection_protocol_VendorExtensionBase.cpp`
- `android-protocol-lib/src/com/google/android/projection/protocol/VendorExtensionBase.java`
- `android-protocol-lib/src/com/google/android/projection/protocol/EchoVendorExtension.java`

## 14.3. VEC sample HU implementation

This section provides pseudocode for a simple HU implementation. It defines the vendor extension in native code; `EchoVendorExtension.java` does the same thing in Java.

First, subclass `VendorExtension` into your own vendor extension class (where `com.vendorname.servicename` is the name used when accessing the VEC):

---

```
class SampleVendorExtension : public VendorExtension {
void SampleVendorExtension::addDiscoveryInfo(ServiceDiscoveryResponse* sdr) {
    Service* service = sdr->add_services();
    service->set_id(id());
    VendorExtensionService* ves = service->mutable_vendor_extension_service();
    ves->set_service_name("com.vendorname.servicename");
    ves->add_package_white_list("com.oem.application.one");
    ves->add_package_white_list("com.oem.application.two");
}
}
```

---

Next, subclass `IVendorExtensionCallbacks` to respond to data received from the MD.

---

```
class SampleVendorExtensionCallbacks : public IVendorExtensionCallbacks {
// This method gets called when the mobile device sends the HU data.
int dataAvailable(uint8_t* data, size_t len) {
// Do something when the data is available.
// For example send it back using sendData.
    mVendorExtension->sendData(data, len);
}
}
```

---

Finally, register your vendor extension like any other service during the initialization sequence.

---

```
galReceiver->init();
... Initialization code ...
VendorExtension* endpoint = new SampleVendorExtension(serviceId,
galReceiver->messageRouter());
SampleVendorExtensionCallbacks* callbacks = new SampleVendorExtension(endpoint);
endpoint->registerCallbacks(callbacks);
galReceiver->registerService(endpoint);
... Other Initialization code ...
galReceiver->start();
```

## 15. Radio

Radio APIs previously documented in HUIG v1.3 should be considered experimental and not included in production HU (R15-010).

## Appendix A: Release Notes

This section provides release notes for the receiver library versions.

### HUIG 2.4

This release includes the following changes.

#### New

- Add support for Touchpad Feedback.
- Add Turn-by-Turn images and information to the Application Status and Data section.
- Add support for different Android Auto UI stream types.

#### Updated

- Update GNSS support and rename the "GPS" section to "GNSS".

#### Removed

- Remove AAP-focused user switching.

### HUIG 2.3.1

This release includes the following changes.

#### New

- Add support for AAP in trucks.

### HUIG 2.3

This release includes the following changes.

#### New

- Add support for Wi-Fi channel negotiation over Bluetooth RFCOMM.
- Add BluetoothAuthenticationResult support for more secure Bluetooth pairing.
- Add support for Touchpad as a primary input method.
- Add support for Electronic Toll Collection (ETC) Icon Display.

### HUIG 2.2.x

This release included the following changes made in HUIG 2.2.0 and 2.2.1.

- Mark the AAP support probe as a beta feature (2.2.1).
- Add support for Wi-Fi channel negotiation (2.2.0).

## HUIG 2.1

This release includes the following changes.

### New

- Add support for wireless projection.
- Add support for user switching.
- Add AAP support probe message.
- Add battery status notification message.
- Add dedicated display name string to `ServiceDiscoveryResponse`.
- Add real DPI support to `VideoConfiguration`.
- Add fuel type information to `SensorSourceService`.
- Add EV charging connector information to `SensorSourceService`.
- Add `CallAvailabilityStatus` message. Allows the HU to indicate when it is ok for AAP to make/receive calls.

### Updated

- Update `SslWrapper` in receiver lib to verify phone certificate name is `CarService`.
- Update receiver lib to be more portable C++.
- Update receiver lib to use boringssl instead of openssl.
- Add experimental fields `ui_absolute` and `tap_as_select` for touchpads.
- Remove return value from `MediaPlayerStatusCallbacks`'s methods.
- Mark haptic support as experimental.
- Replace the ping request callback with a callback that fires only when a bugreport is needed.
- Expose certificate time validation failures from receiver lib to the HU and MD.
- Free mutexes in `SslWrapper`.
- Update android protocol library to handle restarting AAP without unplugging USB.
- Update android protocol library to cleanly shutdown.
- Update android protocol library to send periodic pings.

## Appendix B: Requirements Change Log

This section provides release notes for the HUIG versions.

### HUIG 2.4

This release includes the following changes to requirements.

### New

- To confirm the validity of the HU certificate, the certificate issuing authority and signature are verified, a process that takes place independent of certificate validity dates.

- Restricted **Serial Number** fields are required for the AOAP handshake, but the HU can set them to any value, including the empty string. They are only used to give information to the user if the MD fails to launch Android Auto.
- R04-375 On first-run (the initial AAP connection between an MD and a specific HU), the HU MUST NOT display more than one pop-up from the start trigger until the connection is established. This applies to wired and wireless projection. Once a connection is established, the MD does not send video data until the user grants permission on the MD.
- R05-031 The Wi-Fi network provided by the HU MUST have a unique SSID for each HU.
- R05-055 A Wireless HU MUST NOT advertise more than 21 services in the service discovery phase, since this is the maximum number of UUIDs Android devices can parse at a time.
- R05-510 The HU MUST provide a native screen that enables a user to switch between projection sources.
- R05-511 The HU MUST have the device switching UI accessible in no more than two (2) taps from the AA active session.
- R05-512 The HU MUST retain the link between the Android Auto icon and the current projection session.
- R05-513 The HU SHOULD provide a device switcher widget on the native home screen.
- R05-383 An HU MUST automatically start wireless projection on subsequent connections, if the user sets up AAP over Wi-Fi.
- R05-385 An HU MUST not automatically start wireless projection on subsequent connections, if the user sets up AAP over USB.
- R06-118 The HU MUST keep the AAP session active even if the Bluetooth pairing fails.
- R06-023 The HU MUST allow the user to use Bluetooth on other mobile devices even when AAP is active.
- R07-190 The HU MUST accurately report all fields in **VideoConfiguration**.
- R07-195 The HU MUST report the **real\_density** as defined in the protocol specification.
- R07-281 The reported density by the HU during video setup MUST be added in the **density** field defined in the protocol specification.
- R07-282 The reported density by the HU during video setup MUST be calculated separately for each video resolution supported by the HU.
- R07-283 The reported density by the HU during video setup MUST compensate for the scaling factor by setting the **Video scaling factor** value in the screen size calculator when scaling is used.
- R07-284 The reported density by the HU during video setup MUST be the value resulted in the **Scaled Density** field in the calculator.
- R09-022 If an HU supports multiple touchpads, it MUST declare only one touchpad and combine multiple touchpad inputs on the HU side.
- R09-031 If an HU registers a touchpad, it MAY support haptic and/or acoustic feedback for touchpad as a response to the below feedback events from an MD.
- R09-057 The HU MUST send a corresponding KEYCODE\_DPAD\_CENTER up event to the MD when it detects a mechanical click gesture has completed.
- R09-058 The HU MUST send one KEYCODE\_DPAD\_CENTER up event for every KEYCODE\_DPAD\_CENTER down event.
- The HU MAY use the 'updateServiceDiscovery' message to update those services for which support has already been declared. After service discovery is complete, this message can be sent unsolicited at any time.
- R12-085 GNSS location data MUST support location outputs at a rate of at least 1 Hz.

Restricted .e location data is dead reckoned, data from all sensors used MUST be provided to the MD.

- R12-110 GNSS location data MUST NOT be map matched values.
- R12-120 HUs with GNSS MUST support the LocationCharacterization field in the SensorSourceService declaration.
- R12-125 GNSS location data MUST provide data for all the fields in LocationData (incl. expected horizontal position accuracy (68% radial confidence), altitude, speed and bearing).
- R12-130 GNSS location data MAY be sent to MD when no GNSS position fix is available.
- R12-190 GNSS satellite status SHOULD include the number of GNSS satellites in view.
- R12-203 GNSS satellite status SHOULD report GNSS satellite status as soon as satellites are tracked, even if a location calculated from GNSS is not yet reported.
- R12-205 GNSS satellite status SHOULD report complete GNSS satellite status (i.e. all fields of **GpsSatellite** are valid for each satellite reported) from all constellations tracked (as reported in **GpsSatelliteData**) with the exception of SBAS. The current protocol version only supports GPS satellites, support for other constellations will be added in future revisions.
- The HU SHOULD be able to determine the location in open-sky conditions (strong signals, negligible multipath, HDOP < 2) within 10 seconds (fast time to first fix), when connected to a 0.5 Mbps or faster data speed internet connection. This is typically achieved by the use of Assisted or Predicted GNSS technique to minimize GNSS lock-on time (Assistance data includes Reference Time, Reference Location and Satellite Ephemeris/Clock).
- R12-207 The HU MUST determine its location in open sky within 5 seconds, when location requests are restarted, up to an hour after the initial location calculation, even when the subsequent request is made without a data connection and/or after a power cycle.
- R12-208 The HU MUST be able to determine location within 20 meters and speed within 0.5 meters per second at least 95% of the time in open sky conditions after determining the location.
- R12-209 The HU MUST simultaneously track at least 8 satellites from one constellation.
- The HU SHOULD be able to simultaneously track at least 24 satellites, from multiple constellations (e.g. GPS + at least one of Glonass, Beidou, Galileo).
- R12-223 OEM HUs SHOULD provide 3-axis accelerometer data.
- R12-400 Gyroscope MAY include other axes.
- R13-050 Vehicles that display telephony data in their instrument cluster, heads-up display, etc. for MDs connected over HFP-only MUST display telephony data for phones connected over AAP.

## Updated

- R04-415 When an Android Auto-capable device is connected via cable (USB projection) or RFCOMM (wireless projection), the HU MUST notify the user that Android Auto is available even when another projection technology is active.
- R04-418 If the HU supports multiple smartphone integration technologies that work via USB or Wi-Fi with Android phones, the HU MAY display a screen to enable the user to choose between Android Auto and other available smartphone integration technologies.
- R05-100 Before sending a **WifiStartRequest**, an HU MUST enable its Wi-Fi Access Point and start its TCP server.
- R05-110 If an HU cannot enable its Wi-Fi Access Point without user interaction, it MUST NOT send a **WifiStartRequest** but instead display native guidance to the user on how to setup Wi-Fi.
- R05-445 In case an HU implements a battery-low notification that relies on the battery indicator parameter in the +CIEV result code of the BT HFP profile, the HU MUST suppress this notification for the MD running AAW if the MD has video focus.

- R06-117 The HU MUST drop the Bluetooth pairing and show an error message indicating pairing failure if a **BluetoothAuthenticationResult** message is received with any status other than **STATUS\_SUCCESS** or if 30 seconds have elapsed since the **BluetoothAuthenticationData** message was sent to the MD.
- R07-050 The HU hardware decoder MUST support a minimum frame rate of 30 FPS for all supported video resolutions.
- R07-060 The HU hardware decoder SHOULD support a maximum frame rate of 60 FPS for all supported video resolutions.
- R07-170 The HU screen MUST have a large enough horizontal and vertical resolutions to generate a valid value through the Screen Size Calculator.
- R07-240 The HU MUST support the 1280x720 resolution only if the width is greater than 800 pixels or the height is greater than 480 pixels.
- R07-250 The HU MUST support the 1920x1080 resolution only if the width is greater than 1280 pixels or the height is greater than 720 pixels.
- R12-045 To ensure good performance during navigation, an OEM HU MUST have GNSS.
- R12-070 If the HU does not have GNSS, the vehicle MUST specify a placement location for the MD that allows the MD to achieve optimal GNSS reception considering the constraints of the placement in a vehicle, e.g. indicate a location at the top of the dashboard.
- R12-100 GNSS location data MAY be dead reckoned (DR) values by fusing either the calculated GNSS position or GNSS satellite measurements with additional sensors.
- R12-150 GNSS satellite status MUST accurately report the number of GNSS satellites used in determining the location.
- R12-160 GNSS satellite status MUST have the number of GNSS satellites used in determining the location set to zero (0) when no satellites are being used.
- R12-170 GNSS satellite status MUST have the number of GNSS satellites used in determining the location set to greater than zero (0) otherwise.
- R12-220 OEM HUs MUST provide mechanical speed.
- R12-230 Aftermarket HUs that support only USB projection SHOULD provide mechanical speed, 3-axis accelerometer data and 3-axis gyroscope data.
- R12-390 Gyroscope MUST include all 3-axis gyroscope data.
- R13-001 Vehicles that display navigation data in their instrument cluster, heads-up display, or on any other device from their native navigation solution MUST also display navigation data from AAP.

## Removed

- R05-453 A wireless HU MUST support projection source switching.
- R05-455-B A wireless HU SHOULD not probe for AAP compatibility on MDs connected for the first time even if another AAP session is active with another MD.
- R05-456 A wireless HU MUST NOT trigger projection source switching without user interaction.
- R05-460 The CarConnectedDevices message MUST NOT contain the MD currently projecting.
- R05-470 The CarConnectedDevices message MUST contain an up-to-date list of all nearby paired Bluetooth devices that have successfully completed RFCOMM version negotiation with this HU at least once previously.
- R05-473 The CarConnectedDevices message MUST contain an up-to-date list of all AAP compatible devices connected over USB.
- R05-475 The CarConnectedDevices message MUST be updated to exclude MDs that were unpaired during the

Restricted CarConnectedDevices message MUST be updated to exclude MDs that were unpaired during the active wireless projection session.

- R05-480 The CarConnectedDevices message MUST be sent in response to a CarConnectedDevicesRequest.
- R05-485 The CarConnectedDevices message MUST be sent at least once, within 15 seconds of receiving the request, with the up-to-date list in each message and finalList set to false in all but the last message sent. The finalList MUST be set to true in the last message.
- R05-500 To generate an up-to-date list of nearby devices before sending a CarConnectedDevices message, the HU MUST perform Bluetooth service discovery.
- R05-507 A UserSwitchResponse message sent from a wireless HU MUST have status set to STATUS\_OK if the projection source switching is successful.
- R05-508 The HU MUST also send a ByeByeRequest with ByeByeReason set to DEVICE\_SWITCH.
- R05-509 A UserSwitchResponse message sent from a wireless HU MUST have status set to ERROR\_NO\_RFCOMM\_CONNECTION if the HU cannot establish an RFCOMM connection with the MD selected for device switching.
- R05-520 The list of devices shown in the native screen MUST include all devices generated by the HU when sending the CarConnectedDevices message.
- R05-535 In case the target device (MD2) is connected to the HU over USB and BT RFCOMM, the HU MUST attempt to initialize AAW first upon receiving the 'UserSwitchRequest' from MD1.
- R05-540 When switching to a wired projection source after receiving a UserSwitchRequest, the HU MUST successfully establish an AOAP connection with the target MD before ending the current projection session.
- R05-550 When switching to a wireless projection source after receiving a UserSwitchRequest, the HU MUST successfully establish a TCP session with the target MD before ending the current projection session.
- R05-555 In case of a successful projection source switch, the HU MUST grant unsolicited video focus to the target MD.
- R05-560 If an AAP session with the target device cannot be started due to an error, the HU MUST display a native message reporting the error to the user.
- R05-565 If the error occurs before sending a ByeByeRequest to the current projection session, the HU MUST continue the current projection session.
- R05-570 If the error occurs after sending a ByeByeRequest to the current projection session, the HU MUST return to the native device list.
- R07-180 The HU screen MUST have a vertical height of at least 450dp.
- R12-090 GNSS location data SHOULD be raw GPS values.

## HUIG 2.3.1

This release includes the following changes to requirements.

### New

- R00-010 AAP MAY be integrated into HUs shipped in cars.
- R00-020 AAP MAY be integrated into aftermarket HUs designed for cars.
- R00-030 AAP MAY be integrated into HU shipped in trucks.
- R00-040 If AAP is integrated into a truck, the HU MUST NOT grant navigation focus to the MD when requested.
- R00-050 When an HU in a truck rejects a navigation focus request from the MD, it MUST display a transient native overlay with the following text (or a region appropriate translation of this text): "Android Auto navigation is not available in this vehicle"



- R00-060 If AAP is integrated into a truck, when starting AAP, the HU MUST display a dismissable native disclaimer with the following text (or a region-appropriate translation of this text): "Android Auto navigation apps are not designed for commercial vehicles and may not be used for navigation".
- R00-070 An HU MAY suppress this start-up disclaimer for the second or subsequent connections of a specific MD if the HU receives and remembers per-MD specific user intent to hide this message (through user selection of a 'Do Not Show Again' option, etc.).
- R04-155 Model MUST include the word *truck* at the end of the model string if the HU is shipping in a truck.

## HUIG 2.3

This release includes the following changes to requirements.

### New

- R01-085 Native buttons or affordances that launch Android Auto MUST use the Android Auto Launch Icon.
- R01-115 The Launch Icon MUST NOT be made transparent.
- R01-141 This attribution MUST be displayed as an Android Auto Label icon, an Android Auto text label, OR a combination of the Android Auto Label icon and text label.
- R01-143 If an icon is used, the HU SHOULD use the monochrome Label Icon but MAY use a gray colored icon as a label icon instead of monochrome.
- R01-145 If a textlabel is used, that label MUST be the text string "Android Auto".
- R01-150 If Android Auto icons are resized to match other icons in the native UI, the resizing MUST be performed proportionally.
- R01-160 Android Auto icons MUST NOT be resized smaller than 8mm.
- R01-170 Android Auto icons MUST NOT overlay native content.
- R01-180 Android Auto icons MUST NOT be made transparent.
- R02-003 The integrator's build environment MUST support C++11 to be able to integrate the Receiver Library releases in 2018. Integrators should expect that C++17 support will be required from 2019.
- R05-025 Wireless HUs MUST support 5 GHz 802.11ac.
- R05-038 The Wi-Fi network provided by the HU MUST assign only IPv4 addresses for wireless projection. IPv6 is not currently supported.
- R05-086 The `WifiVersionRequest` MAY include one 2.4 GHz frequency and one 5 GHz frequency if the access point broadcasts on both bands simultaneously.
- R05-089 After receiving a `WifiVersionResponse` message with `version_status` set to `STATUS_NO_SUPPORTED_WIFI_CHANNELS`, the HU MAY send another `WifiVersionRequest` with a different Wi-Fi channel frequency to find a frequency supported by the MD.
- R05-265 If the MD does not connect to the HU's access point within 60 seconds of receiving the `WifiStartRequest`, the HU MAY display a message asking the user to restart Bluetooth on the MD.
- R05-315 If the HU does not receive a TCP connection request from the MD within 45 seconds, the HU MAY show a popup prompting the user to restart Bluetooth and Wi-Fi on the MD.
- R07-500 Each supported resolution MUST be reported with the maximum frame rate applicable.
- R07-510 To ensure high resolutions are utilized, the HU MUST advertise the supported video configurations with the highest resolution first.
- R08-175 When the HU is ducking native media and mixing in transient audio from the MD, the HU MUST ensure

Restricted  
that transient audio from the MD is clearly audible to the driver.

- R08-435 The HU MUST maintain/remember volume settings for each stream and not reset them between Android Auto sessions.
- R08-485 The HU MUST remain in the mute state after a phone call ends.
- R08-561 Any affordance that implements this functionality SHOULD have a "mute/pause" (or equivalent) label.
- R09-026 If an HU supports a touchpad, it MUST declare a touchpad as an input device.
- R09-027 When registering a touchpad, the HU MAY set the `ui_navigation` flag to false.
- R09-028 When a touchpad is registered with the `ui_navigation` flag set to false, the HU MUST also register either touchscreen or rotary controller.
- R09-029 When registering a touchpad, the HU MAY set the `ui_navigation` flag to true.
- R09-030 If an HU registers a touchpad for AAP, it MUST send all touchpad events with absolute position values to the AAP receiver when video focus is Projection.
- R09-040 Values for multiple fingers MAY be sent.
- R09-041 An HU MUST specify the sensitivity of a touchpad during service discovery.
- R09-042 An HU MAY update its sensitivity while an AAP session is active by calling ReceiverLib function `InputSource::updateTouchPadUiSensitivity()` with the new sensitivity setting as its parameter and then calling `GalReceiver::updateService()` with the InputSource object as its parameter.
- R09-045 An HU MUST specify a sensitivity setting between 1 and 10.
- R09-050 If an HU registers a touchpad, it MUST have a mechanical click mechanism on the touchpad to trigger the select action.
- R09-055 The HU MUST send `KEYCODE_DPAD_CENTER` to the MD when it detects a mechanical click event on the touchpad.
- R09-065 An HU MUST NOT register a touchpad with the `ui_navigation` flag set to true and a rotary controller at the same time.
- R12-600 An HU MAY report a `SENSOR_TOLL_CARD` sensor during Service Discovery.
- R12-610 If `SENSOR_TOLL_CARD` sensor is reported, the HU MUST report the current state of an ETC card (true: present, false: not present) when the state is changed.
- R12-620 `SENSOR_TOLL_CARD` sensor MUST only be reported in HUs shipping in Japan.

## Updated

- R01-140 Native widgets or screens in the HU MUST show some form of Android Auto attribution on all content originating from or pertaining to Android Auto.
- R04-280 Head Unit Software Build MUST be set to "multi" (all lower case letters) as the value if HU does not have specific build numbers.
- R05-035 The HU MUST provide a native user interface to allow the user to automatically generate a new PSK and/or manually change the current PSK.
- R05-036 The Wi-Fi network provided by the HU MUST be advertised as 'ANDROID\_METERED' over the vendor-specific option 43 in the access point DHCP configuration if the access point has Internet access.
- R05-085 The `WifiVersionRequest` MUST include the frequency of the Wi-Fi channel on which the HU's access point intends to broadcast.
- R05-088 The HU MUST NOT send a `WifiStartRequest` until it has received a `WifiVersionResponse` message with `version_status` set to SUCCESS.

- R05-360 A wireless HU MUST NOT attempt to start both wired and wireless projection with the same device and MUST NOT attempt to switch an MD to AOA mode while it is projecting over wireless.
- R06-115 The HU MUST confirm the Bluetooth pairing when a BluetoothAuthenticationResult message is received with STATUS\_SUCCESS.
- R06-116 The HU MUST NOT confirm the Bluetooth pairing until a BluetoothAuthenticationResult message is received with STATUS\_SUCCESS.
- R06-117 The HU MUST drop the Bluetooth pairing and show an error message indicating pairing failure if a BluetoothAuthenticationResult message is received with any status other than STATUS\_SUCCESS or if 60 seconds passed since the BluetoothAuthenticationData message was sent to the MD.
- R06-120 If the HU receives a BluetoothPairingRequest when a separate already connected MD is engaged in a Bluetooth HFP call, the HU MUST send a BluetoothPairingResponse with status set to STATUS\_BLUETOOTH\_PAIRING\_DELAYED if it is unable to pair with a new MD during an active call.
- R07-110 If an HU has a touchscreen, the AAP display area MUST be at least 80mm high excluding any screen area covered by a bezel.
- R07-120 If an HU has a touchscreen, the AAP display area MUST be at least 132mm wide excluding any screen area covered by a bezel.
- R07-240 The HU MUST support the 1280x720 resolution if the physical resolution of the projection content area is large enough to display 1280x720 video.
- R07-250 The HU MUST support the 1920x1080 resolution if the physical resolution of the projection content area is large enough to display 1920x1080 video.
- R07-440 In response to a video focus request, the HU MUST send a video focus notification to the MD, indicating the new video focus state, even if the video focus state remains unchanged.
- R08-012 For HU implementations that support AAW, the HU MUST support both the PCM codec and AAC-LC codec. To do this, the HU registers two audio sinks per device-to-vehicle audio channel, one for PCM and one for AAC-LC.
- R08-460 If AAP is being displayed full-screen, volume MUST be shown in an overlay.
- R08-480 In Global Mute, all audio sources are muted. If the HU implements Global Mute it MUST NOT play any audio from the MD when muted, except phone call Audio.
- R08-560 If the HU implements source-specific mute and the current source is AAP media, the HU MAY stop AAP media playback when the HU is muted; for example, the HU MAY send KEYCODE\_MEDIA\_PAUSE to the MD to pause AAP media and KEYCODE\_MEDIA\_PLAY to resume AAP media.
- R09-010 The HU MUST declare one of the following supported input device combinations in Service Discovery: Touchscreen only, Rotary only, Touchpad only, Touchscreen and rotary, Rotary and touchpad, Touchscreen and touchpad.
- R09-260 KEYCODE\_MEDIA\_PLAY\_PAUSE. If the HU has a hardkey that plays and pauses media, the HU must send KEYCODE\_MEDIA\_PLAY\_PAUSE to the MD if the button is pushed when AAP has audio focus.
- R09-261 KEYCODE\_MEDIA\_PLAY. If the HU has a hardkey Play button, the HU MUST send KEYCODE\_MEDIA\_PLAY to the MD if the button is pushed when AAP has audio focus.
- R09-262 KEYCODE\_MEDIA\_PAUSE. If the HU has a hardkey Pause button, the HU MUST send KEYCODE\_MEDIA\_Pause to the MD if the button is pushed when AAP has audio focus
- R09-263 KEYCODE\_MEDIA\_PREVIOUS. If the HU has a hardkey Previous button, the HU MUST send KEYCODE\_MEDIA\_PREVIOUS to the MD if the button is pushed when AAP has audio focus.
- R09-264 KEYCODE\_MEDIA\_NEXT. If the HU has a hardkey Next button, the HU MUST send KEYCODE\_MEDIA\_NEXT to the MD if the button is pushed when AAP has audio focus.

## Removed

- R08-265 If the MD was the last active media source, the HU must send an unsolicited audio focus notification to restore focus to the MD.
- R13-036 If the HU's media widget is selectable, selecting the widget MUST send KEYCODE\_MEDIA to the MD to show the AAP media facet.

## HUIG 2.2.x

This release includes the following changes to requirements made in HUIG 2.2.0 and 2.2.1.

### General Guideline Updates

- 5.8.3. Projection source switching scenarios (switching from wired projection to wired projection).
- 8.3.2. Audio focus notifications (MD behavior).
- 8.3.3. Audio focus implementation details.
- 5.2 Wireless connection process overview (Bluetooth HFP connection).

### New

- R01-095 When an AAP session is active, the Launch Icon MUST be placed directly on the home screen to enable the user to launch Android Auto in one gesture.
- R03-205 The HU MUST be able to probe for AAP compatibility as soon as an MD is connected over USB provided it has no active AAW session with the same MD.
- R03-210 The probe MUST be performed even if the HU already has an active AAP session with another MD.
- R03-015 The HU MUST not support AOA V2.0.
- R03-177 If the disconnect message is received, the HU re-enumerates USB and MUST not automatically restart AAP.
- R03-180 A ByeByeRequest from the MD includes a ByeByeReason and the HU MUST not allow the user to start AAP unless the MD is unplugged/replugged if the ByeByeReason is 'NOT\_SUPPORTED' or 'NOT\_CURRENTLY\_SUPPORTED' ().
- R03-185 A ByeByeRequest from the MD includes a ByeByeReason and the HU MUST allow the user to start AAP without unplugging/replugging the MD if the ByeByeReason is 'USER\_SELECTION', 'DEVICE\_SWITCH' or 'PROBE\_SUPPORTED'.
- R03-187 A ByeByeRequest from the MD includes a ByeByeReason and the HU MUST MUST not mark an MD as AAP incompatible for future connections unless the ByeByeReason is 'NOT\_SUPPORTED'.
- R04-418 If the HU supports multiple smartphone integration technologies that work via USB with Android phones, the HU MAY display a screen to choose between Android Auto and other available smartphone integration technologies when an Android phone is connected for the first time.
- R04-490 In case of SSL authentication failure, the ReceiverLib 'IControllerCallbacks::unrecoverableErrorCallback' is called with errorcode 'STATUS\_AUTHENTICATION\_FAILURE\_CERT\_NOT\_YET\_VALID'.
- R05-032 The Wi-Fi network provided by the HU MUST use Wi-Fi Protected Access II (WPA2) encryption (RSN/AES) to ensure the security of the connection to the MD.
- R05-033 The Wi-Fi network provided by the HU MUST be protected by a unique Pre-Shared Key (PSK) generated randomly using a Cryptographically Secure Pseudo Random Number Generator (CSPRNG) upon the access point's first initialization.
- R05-034 The generated key MUST be at least 12 characters long and formed of the set of characters that are upper case, lower case and numbers.

- R05-035 The HU MUST provide a native user interface to allow the user to manually change the PSK.
- R05-036 The Wi-Fi network provided by the HU MUST be advertised as 'ANDROID\_METERED' over the vendor-specific option 43 in the access point DHCP configuration to enable the MD to suppress data exhaustive activities while AAW is active.
- Step 3 in section 5.2 (Bluetooth HFP connection).
- R05-075 An MD that supports wireless projection may initiate a RFCOMM connection to a wireless HU before or after a Bluetooth HFP connection is established and the wireless HU MUST be able to handle both scenarios.
- R05-077 The HU MUST establish and maintain the RFCOMM connection even if a wired projection session is started and/or the Bluetooth connection itself is triggered by AAP as part of the AAP connection and launch process.
- R05-085 The HU MUST also inform the MD which Wi-Fi channel frequencies are supported by the HU's access point.
- R05-088 If the HU has received a STATUS\_NO\_SUPPORTED\_WIFI\_CHANNELS from the MD, the HU MUST not send a WifiStartRequest.
- R05-125 An MD may also send a WifiStartRequest to start wireless projection and the wireless HU MUST attempt to start AAW as a result.
- R05-165 A WifiStartResponse message sent from a wireless HU MUST have status set to STATUS\_WIFI\_DISABLED if Wi-Fi is disabled on the HU and cannot be enabled automatically.
- R05-235 If the wireless HU has two access points with the same SSID on different frequency bands, the BSSID of the 5 GHz access point MUST be utilized.
- R05-395 The Logic to track Ping RTT is not part of the receiver library and MUST be implemented on the HU side.
- R05-405 If the HU receives a ByeByeRequest with the reason being 'USER\_SELECTION' or DEVICE\_SWITCH, the HU MUST return to a native screen and be able to restart wireless projection without disconnecting the Wi-Fi connection or the Bluetooth connection with the MD.
- R05-407 A wireless HU MUST not end the active wireless projection session even if the Bluetooth connection with the projecting MD is disabled.
- R05-445 In case an HU implements a battery-low notification that relies on the battery indicator parameter in the +CIEV result code of the BT HFP profile, the HU MUST suppress this notification for the MD running AAW during the active session.
- R05-455 An HU that supports projection source switching: MUST probe for AAP compatibility on MDs connected for the first time even if another AAP session is active with another MD.
- R05-456 An HU that supports projection source switching MUST NOT trigger projection source switching without user interaction.
- R05-473 The CarConnectedDevices message MUST contain an up-to-date list of all AAP compatible devices connected over USB.
- R05-475 The CarConnectedDevices message MUST be updated to exclude MDs that were unpaired during the active wireless projection session
- R05-480 The CarConnectedDevices message MUST be sent in response to a CarConnectedDevicesRequest
- R05-485 The CarConnectedDevices message MUST be sent at least once, within 15 seconds of receiving the request, with the up-to-date list in each message and finalList set to false in all but the last message sent. The finalList MUST be set to true in the last message.
- R05-507 A UserSwitchResponse message sent from a wireless HU MUST have status set to STATUS\_OK if the projection source switching is successful.

- Restricted HU MUST also send a ByeByeRequest with ByeByeReason set to DEVICE\_SWITCH.
- R05-509 A UserSwitchResponse message sent from a wireless HU MUST have status set to ERROR\_NO\_RFCOMM\_CONNECTION if the HU cannot establish an RFCOMM connection with the MD selected for device switching.
- Switching from wired projection to wired projection in 5.8.3
- Labels for Figure 5.7 & 5.8
- R05-555 In case of a successful projection source switch, the HU MUST grant unsolicited video focus to the target MD.
- R05-560 If an AAP session with the target device cannot be started due to an error, the HU MUST display a native message reporting the error to the user.
- R05-565 If the error occurs before sending a ByeByeRequest to the current projection session, the HU MUST continue the current projection session.
- R05-570 If the error occurs after sending a ByeByeRequest to the current projection session, the HU MUST return to the native device list.
- R06-021 The HU MUST NOT require user interaction during the Bluetooth pairing and connection process (except in the case where the upper limit to the number of supported paired Bluetooth devices has been reached).
- R06-085 The HU MUST send a BluetoothPairingResponse with STATUS\_SUCCESS as the status if a BluetoothPairingResponse with STATUS\_BLUETOOTH\_PAIRING\_DELAYED was sent previously and the HU is now ready to pair.
- R08-012 For HU implementations that support wireless AAP, the HU MUST support both the PCM codec and AAC-LC codec
- updates to 8.3.2 on MD behavior 'Audio focus notifications'
- updates to 8.3.3. 'Audio focus implementation details'
- R08-265 If the MD was the last active media source, the HU must send an unsolicited audio focus notification to restore focus to the MD.
- R08-408 The HU MUST NOT adjust the output of the MD audio through ducking or boosting aside from user initiated volume changes.
- R11-031 an HU that includes native navigation MUST allow native maps to be viewed when an AAP session is active, regardless of navigation focus mode.
- R11-032 an HU that includes native navigation MUST allow native navigation to be started when an AAP session is active.

## Updated

- R03-010 The HU MUST be a USB 2.0 Hi-Speed Host or USB 3.0 Host and provide host support for the Android Open Accessory (AOA) protocol V1.0.
- R03-160 If no response to the Ping message is received within three (3) seconds, the HU MUST show a message to the user informing the user that AAP has stopped, and there is a problem with the connection.
- R05-400 If Ping RTT exceeds three (3) seconds when a wireless projection session is active (including the case when a PingResponse is never received), the HU MUST display a native message stating that the MD has disconnected.
- R07-050 The HU hardware decoder MUST support a minimum frame rate of 30 FPS
- R09-227 KEYCODE\_CALL SHOULD not be used. The Bluetooth HFP 'ATA' command MUST be used for answering incoming calls.



- Restricted CODE\_ENDCALL SHOULD not be used. The Bluetooth HFP AT+CHUP command MUST be used for rejecting incoming calls as well as ending active calls.
- R10-285 Incorrect branding of Android Auto's ASR features such as "Google Voice", "Google Voice Search", "Google Search", "Google voice actions", and "Android Auto voice actions" are not compliant with Google branding and MUST NOT be used.

## HUIG 2.1

This release includes the following changes to requirements.

### New

This version of the HUIG includes the following new requirements:

- R01-035 A wireless HU MUST also show the Launch Icon in the available state when an MD is available to start wireless projection (Bluetooth RFCOMM connection and version negotiation are complete).
- R01-131 If a projection session is active, the HU MUST grant video focus to the MD.
- R01-132 If a projection session is not active, and the HU does not support wireless projection, the HU MUST display a native message informing the user to connect an Android phone to the car with a USB cable to start using Android Auto.
- R01-133 If a projection session is not active, the HU does support wireless projection and no MDs are available for wireless projection, the HU MUST display a native message informing the user to connect an Android phone to the car with a USB cable or to pair an Android phone via Bluetooth to start using Android Auto.
- R01-134 If a projection session is not active, the HU does support wireless projection and one MD is available for wireless projection, the HU MUST send a **WifiStartRequest** to that MD.
- R01-135 If a projection session is not active, the HU does support wireless projection and multiple MDs are available for wireless projection, the HU MUST show a native screen listing all MDs available for wireless projection.
- R03-190 If an HU that supports wireless AAP has completed RFCOMM version negotiation with an MD, but has never had a wireless session with that MD, after concluding a wired session, the HU SHOULD display native guidance to the user that they can connect wirelessly with their MD the next time.
- R03-200 If the probe session indicates that AAP is supported, the HU MUST notify the user that Android Auto is available and provide an option to start Android Auto.
- R04-075 If an HU supports wireless projection, it MUST register a **WifiProjectionEndpoint** service.
- R04-360 MUST be consistent across Android Auto compatible products.
- R04-370 MUST be UTF-8.
- R04-415 The HU MUST notify the user of the availability of Android Auto on USB connection of an Android Auto capable device.
- R04-470 In this case, the HU MUST inform the user about the reason for the SSL authentication failure and provide guidance on how to resolve the issue.
- R04-480 The HU MUST inform the user that Android Auto is available and enable the user to start a new Android Auto connection.
- R05-010 If an HU supports AAP protocol v1.4 or later and meets the wireless hardware and latency requirements defined in this section, then it MUST support wireless projection and wired (USB) projection.
- R05-020 If an HU supports AAP Protocol v1.3 or earlier or does not meet the wireless hardware and latency requirements, then it MUST support only wired projection.
- R05-030 Wireless HUs MUST provide a wireless Local Area Network (Wi-Fi) with the HU as the access point.



- Restricted The connection over Wi-Fi between a wireless HU and MD MUST support a sustained throughput of at least 4 MBps while maintaining a Round Trip Time (RTT) of less than 200ms.
- R05-050 To indicate that it supports wireless projection, a wireless HU MUST advertise the Android Auto UUID `4de17a00-52cb-11e6-bdf4-0800200c9a66` at all times it participates in Bluetooth Service Discovery.
- R05-060 A wireless HU MUST support version 1.0 of the Bluetooth Radio frequency communication (RFCOMM) protocol and implement a Bluetooth RFCOMM server.
- R05-070 A wireless HU MUST support an RFCOMM connection over each active Bluetooth connection.
- R05-080 A wireless HU MUST send a `WifiVersionRequest` to the MD immediately after RFCOMM connection (within 1s) informing the MD the highest version of the wireless projection RFCOMM control channel it implements.
- R05-090 A `WifiStartRequest` sent from HU to MD MUST contain the IP address and port number for the TCP connection that follows from the MD to the HU.
- R05-100 Before sending a `WifiStartRequest`, an HU MUST enable its Wi-Fi Access Point.
- R05-110 If a HU cannot enable its Wi-Fi Access Point without user interaction, it MUST not send a `WifiStartRequest` but instead display native guidance to the user on how to setup Wi-Fi.
- R05-120 A wireless HU MAY implement a mechanism to automatically start wireless projection (by sending a `WifiStartRequest`) for devices that have been previously connected, etc.
- R05-130 If the HU does not receive a `WifiStartResponse` within 25 seconds of sending a `WifiStartRequest`, the HU MUST display a native error message to the user informing them that the MD could not start wireless projection successfully.
- R05-140 MUST be sent immediately (within 1s) of receiving a `WifiStartRequest`.
- R05-150 MUST contain an IP address and port number for the TCP connection that follows from the MD to the HU.
- R05-160 MUST have status set to `STATUS_PROJECTION_STARTED_ALREADY` if a projection session is already active between the HU and any device.
- R05-170 MUST have status set to `STATUS_WIFI_NOT_YET_STARTED` if the HU is able to enable Wi-Fi but has not yet completed this task.
- R05-180 MUST have status set to `STATUS_SUCCESS` if the HU is ready to proceed to next step of the connection process.
- R05-190 If a HU sends a `WifiStartResponse` message with status set to `STATUS_WIFI_DISABLED`, it must also display native guidance to the user on how to setup Wi-Fi.
- R05-200 A wireless HU MUST support exchange of Wi-Fi credentials via both the AAP protocol as part of Service Discovery and via Bluetooth RFCOMM communication.
- R05-210 MUST specify a `wifi_ssid` as an ASCII string.
- R05-220 MUST specify a `wifi_password` as an UTF-8 string.
- R05-230 MUST specify a `wifi_bssid` in Ethernet MAC address format.
- R05-240 MUST be sent immediately (within 1s) after receiving a `WifiInfoRequest`.
- R05-250 If the HU receives a `WifiConnectStatus` with status of `STATUS_WIFI_INACCESSIBLE_CHANNEL` it MAY display a native error message giving instructions to the user on how to fix or troubleshoot the issue.
- R05-260 If a read error occurs in the HU Bluetooth RFCOMM socket, the HU MUST wait for the MD to reconnect RFCOMM and then send the MD a `WifiVersionRequest`.
- R05-270 The Bluetooth RFCOMM connection MUST NOT be ended for the duration of the Bluetooth HFP connection between the HU and MD, even if a projection session has ended.

Restricted between the HU and MD, even if a projection session has ended.

- R05-280 A wireless HU MUST implement a TCP server.
- R05-290 An HU MUST be able to accept incoming TCP connection from a second device while a projection (wireless or wired) is active with a first device to enable seamless wireless projection user switching.
- R05-300 An HU MUST be able to end a projection session without shutting down the TCP server.
- R05-310 The TCP server MUST set the following connection socket settings.
- R05-320 If the HU has not sent or received a **ByeByeRequest** to/from the MD and detects a TCP socket read timeout or receives a reset (RST) packet, the TCP server MUST close its open socket and the HU MUST display a native message stating that the MD has disconnected.
- R05-330 If the HU receives a **ByeByeRequest** from the MD, the TCP server on the HU MUST close its open socket only after it has sent a **ByeByeResponse** message.
- R05-340 After sending a **WifiStartRequest** to an MD, a wireless HU MUST NOT send a second **WifiStartRequest** request to a different MD before it receives a **WifiStartResponse** from the first MD.
- R05-350 If a wireless HU receives multiple concurrent **WifiStartRequest** messages from different MDs, it MUST send a **WifiStartResponse** with status set to **STATUS\_PROJECTION\_STARTED\_ALREADY** to the MD(s) that it does not intend to start projection with.
- R05-360 A wireless HU MUST NOT attempt to start both wired and wireless projection with the same device.
- R05-370 MUST NOT send a **WifiStartRequest** when wired projection is already .active or in the process of setting up.
- R05-380 A wireless HU must send a **PingRequest** message to the MD every second.
- R05-390 and MUST measure and track *ping round-trip-time (Ping RTT)* as the elapsed time between sending a **PingRequest** message and receiving a **PingResponse**.
- R05-400 If Ping RTT exceeds five (5) seconds when a wireless projection session is active (including the case when a **PingResponse** is never received), the HU MUST display a native message stating that the MD has disconnected.
- R05-410 If Ping RTT is above 200ms for five (5) consecutive **PingRequest** messages, the HU MUST display a native message stating that Wi-Fi network connectivity is poor and asking the user to improve placement of the MD.
- R05-420 If an HU has video focus set to native and receives a **BatteryStatusNotification** with **critical\_battery** set to TRUE, the HU MUST display a native message stating that the MD has low battery and should be plugged in or charged immediately.
- R05-430 This native message SHOULD include the current battery level and estimated time remaining in seconds.
- R05-440 If a HU has video focus set to projected and receives a **BatteryStatusNotification** with **critical\_battery** set to TRUE, the HU MUST NOT display a native message as the Android Auto UI will display a message.
- R05-450 An HU MUST NOT support support multiple concurrent projection sessions.
- R05-460 MUST NOT contain the MD currently projecting.
- R05-470 MUST contain an up-to-date list of all nearby paired Bluetooth devices that have successfully completed RFCOMM version negotiation with this HU at least once previously.
- R05-500 To generate an up-to-date list of nearby devices before sending a **CarConnectedDevices** message, the HU MUST perform Bluetooth service discovery.
- R05-510 If multiple MDs are available for projection (including any devices actively projecting), a wireless HU MUST provide a native screen that enables a user to switch projection sources

- R05-520 The list of devices shown in the native screen MUST include all devices generated by the HU when sending the **CarConnectedDevices** message
- R05-530 The HU MUST use the most preferred name available when showing a device in a native screen.
- R05-540 When switching to a wired projection source after receiving a **UserSwitchRequest**, the HU MUST successfully establish an AOAP connection with the target MD before ending the current projection session.
- R05-550 When switching to a wireless projection source after receiving a **UserSwitchRequest**, the HU MUST successfully establish a TCP session with the target MD before ending the current projection session.
- R05-560 If an AAP session with the target device cannot be started due to an error that occurs after sending a **ByeByeRequest** to the current projection session, the HU MUST display a native message reporting the error to the user and then return to the native device list.
- R06-025 An HU that supports wireless projection MUST support multiple simultaneous Bluetooth HFP connections to enable seamless wireless projection user switching.
- R06-240 An HU MAY allow a phone call to be made from a second connected HFP device or an embedded SIM when an AAP session is active.
- R06-250 When starting a call from a second connected HFP device or an embedded SIM, the HU MUST call the Receiver Library **CallAvailabilityStatus** .message with **call\_available** set to **false**.
- R06-260 When ending a call from a second connected HFP device or an embedded SIM, the HU MUST call the Receiver Library **CallAvailabilityStatus** message with **call\_available** set to **true**.
- R06-270 When starting an AAP session while on a call with a second connected HFP device or an embedded SIM, the HU MUST send the MD a **CallAvailabilityStatus** message with **call\_available** set to **false** immediately after completing AAP Service Discovery.
- R12-045 To ensure good performance during navigation, the HU SHOULD have GPS.
- R12-050 HUs that support wireless projection MUST have GPS.
- R12-491 Fuel Type SHOULD be provided by the HU.
- R12-492 Fuel Type if sent, MUST include all fuel types the vehicle supports.
- R12-493 EV Connector Type MUST be sent if an HU sends a Fuel Type of **FUEL\_TYPE\_ELECTRIC**.
- R12-494 EV Connector Type MUST be sent in order of preference, for example prefer fast charge.

## Updated

This version of the HUIG includes updates to the following existing requirements:

- R03-150 The HU MUST send a PingRequest message to the MD every second.

## Administrative

This version of the HUIG includes minor corrections (grammatical, typo fixes, etc.) to the following existing requirements:

- R01-040 HUs distributed in a countries where AAP has launched SHOULD show the Launch Icon in the unavailable state when an MD is not connected or a previous AAP session has been exited by the user.
- R01-050 HUs distributed in countries where AAP has not launched MUST NOT show the Launch Icon unless an AAP session is active or an MD is available to start wireless projection.
- R04-430 The HU SHOULD give the MD video focus upon connection without requiring the user to initiate any action.
- R12-060 If the HU has GPS, it MUST send GPS location data to the MD.

- **Restricted** If the HU does not have GPS, the vehicle MUST specify a placement location for the MD such that the performance of the MD GPS is equivalent to performance outside of the vehicle.

---











Was this article helpful?

Yes

No

---

## Help

-  Overview
-  **Head Unit Integration Guide**
-  HMI Integration Guide
-  Receiver Library APIs
-  Protocol Specification
-  Experiment Methodology
-  Google Assistant Driving Mode
-  WLAN Connectivity
-  High Resolution and Widescreen
-  Vehicle Make Logo Exit Icon

