sdl

SDL Overview Guides
Document current as of 02/09/2022 11:52 AM.

Introduction

This is an overview of how SmartDeviceLink (SDL) works. It will go through higher level discussions about certain aspects of SDL to give an intro level of understanding. For more general information on SmartDeviceLink (SDL) as a technology, there is a white-paper available here.

SmartDeviceLink is a connectivity protocol and set of associated libraries that allow third-party applications to communicate with cars. This allows drivers to safely access the content they want through their vehicle.

There are different components to SDL that make everything work. The following diagram shows how some of those pieces are connected.

SmartDeviceLink High Level Diagram
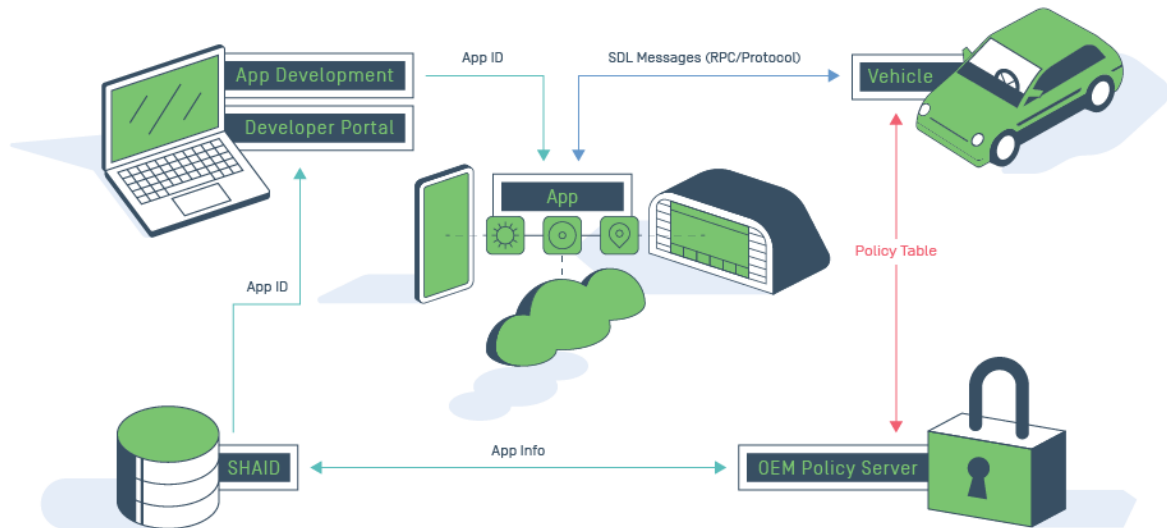
Sections

Supported Platforms

This section quickly points out which platforms can be used for the application, Core, and HMI parts of the project.

Getting Started

The first few guides will be centered around getting SDL started.

1. Establish a connection over the transport.
2. How the app starts the SDL protocol and how Core accepts.
3. How the app registers itself onto the head unit and how Core responds with system information.
4. How the app is activated for user interaction

Protocol Spec

This section describes the lowest level of SDL. It defines how the byte level data is communicated between the application library and the head unit / Core. It contains information on what protocol level packets are used, basic handshakes, etc.

RPC Spec

This section describes the higher level JSON-RPC messages that get sent. These messages are higher-level messages sent using the protocol spec's lower level messages (see above). The majority of messages that get sent between the app and head unit are RPC messages. The RPC messages contain information to perform and respond to actions, notify of events, etc. The RPC spec defines much of what SDL can do.

RPC Spec Generation

This section describes how the SDL platforms use a code generating script to create RPC classes from the RPC Spec.

Policies

SmartDeviceLink uses policies to enforce rules about which messages can be sent to and from connected apps. This section will give an overview of that functionality and how to update the policy table, which declares these policies for all known apps.

App Services

Connected applications can augment a head unit's service offerings (such as navigation and weather) using SDL. Head units can feed the service specific information into their own system as a unified API for that app service type. This includes displaying service data outside the app's template view and makes that app feel integrated directly with the head unit.

Best Practices

This section contains items that aren't hard requirements but are best practices to ensure that the widest variety of apps work properly with your head unit.

FAQ

This section covers a few of the commonly asked questions around SDL.

Supported Platforms

App Platforms

There are currently several supported platforms for integrating SDL into existing applications or creating new ones specifically for SDL.

ANDROID

The Android SDL Library is specifically for creating applications that run on Android mobile versions of the operating system.

IOS

The iOS SDL Library is specifically for creating applications that run on iOS mobile versions of the operating system.

JAVA SE

The standard version of Java is supported to be used mostly embedded or local on the same hardware or network as the Core implementation. It can be ran from a backend as

well, however, handling many connections at a time would be something that is handled outside of the Java SE SDL library.

JAVA EE

The enterprise edition of Java is supported, however, there is no Java EE specific code in the library itself. This is due to the fact that Java EE has conflicts with our licensing model for the open source library. This library is designed to be ran from a backend leveraging Java Beans and other web specific Java features.

JAVASCRIPT

The vanilla JavaScript Library allows you to create applications that run in the browser. WebEngine applications, which are installed via an app store and are run by supported HMIs, also fall into this category.

NODE.JS

The Node.js Library allows you to create applications that run on a web server. These applications can also take advantage of npm packages.

Core Platforms

UBUNTU

Core currently officially supports Ubuntu versions 18.04 and 20.04. Ubuntu 20.04 is the main supported platform of the project.

QNX

Many adopters have built Core on the QNX platform. It does require a decent amount of work. This will include switching out common linux based libraries with the QNX specific version or creating one from scratch.

ANDROID EMBEDDED

Some adopters have built Core on the Android platform. It does require a decent amount of work. This will include switching out common linux based libraries with the Android specific version or creating one from scratch.

HMI Platforms

The HMI can be written in any platform as long as it can connect through websockets to Core and use the HMI spec as defined with the version of Core that is being used.

HTML

HTML is the most widely supported platform for developing the HMI. There are several examples of using HTML in the open, and the best example is the Generic HMI.


QT

Some partners have created HMI implementations using QT. This platform is usable, but is a bit less dynamic and more difficult to update than HTML.

Establishing a Transport

SDL is theoretically transport agnostic, meaning it can work over any potential transport. However, due to limitations of app platforms, only certain transports are available for connecting to given app platforms.

The first step in using SDL is to establish a connection over a primary transport.
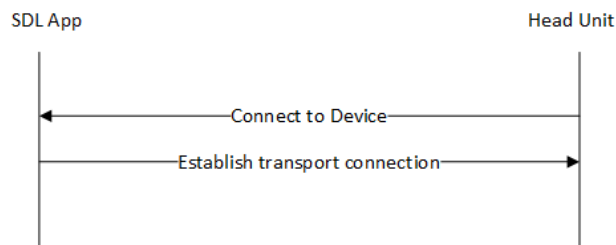
Primary and Secondary Transports

SDL has the ability to register a session across multiple transports. This is useful for certain scenarios where a higher bandwidth transport is needed to perform more data intensive operations of SDL (such as video/audio streaming), but that transport isn't good for establishing or maintaining the connection.
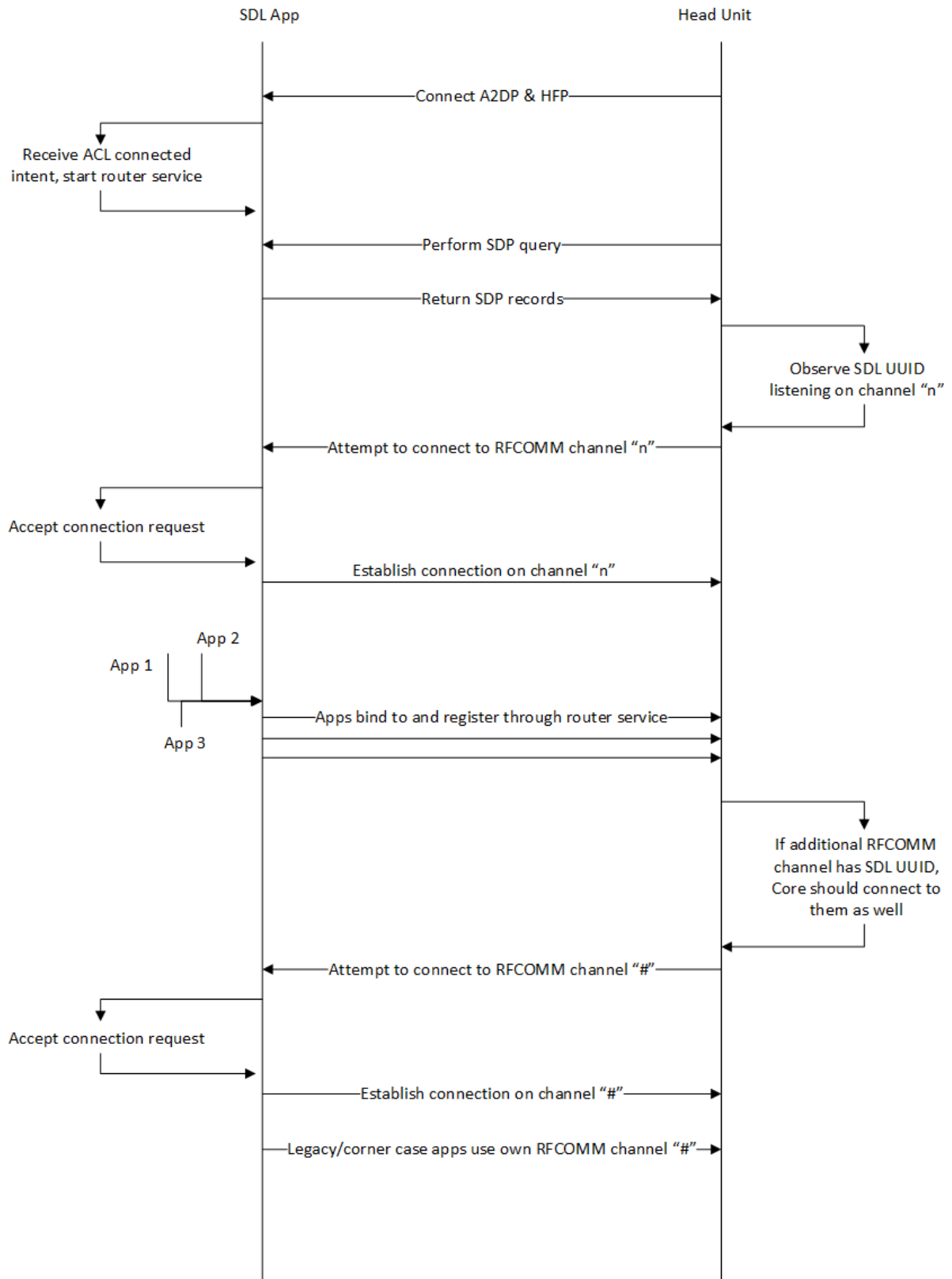
Android

BLUETOOTH

Primary or secondary transport

Bluetooth connections established to an Android device require a few prerequisites and processes. First, the bluetooth's Advanced Audio Distribution Profile (A2DP) and Hands Free Profile (HFP) must be connected for the SDL Android router service to start listening on an RFCOMM channel using the Serial Port Profile (SPP). The Android device will use a router service that allows multiple apps to bind to it and utilize a single transport connection. This allows for a greater chance that the SDL library can obtain one of the

RFCOMM channels available on the device, as most devices have rather low limits. In some cases, apps might not bind to the router service and listen on their own bluetooth transport. These apps must also be connected.

SDL App   Head Unit

←————————————— Connect A2DP & HFP ——————————————

Receive ACL connected
intent, start router service

←————————————— Perform SDP query ———————————————

——————————————— Return SDP records ——————————————→

                                        Observe SDL UUID
                                        listening on channel "n"

←———————— Attempt to connect to RFCOMM channel "n" ————

Accept connection request

——————————— Establish connection on channel "n" ————————→

      App 2
App 1

                ——— Apps bind to and register through router service ——→

      App 3

                                        If additional RFCOMM
                                        channel has SDL UUID,
                                        Core should connect to
                                        them as well

←———————— Attempt to connect to RFCOMM channel "#" ————

Accept connection request

——————————— Establish connection on channel "#" ————————→

——— Legacy/corner case apps use own RFCOMM channel "#" ——→

CONNECTION TIMING

- After the initial connection of A2DP and HFP, the head unit will only have a limited time to connect to the SDL RFCOMM channel. This is due to several Android operating system restrictions and requirements.
- The router service must be started as a foreground service which causes a notification to appear on the users phone. The library makes adjustments to the time the router service spends in the foreground based on three scenarios:

  1. **The router service has never seen this hardware before:** In this case the router service will start in the foreground and remain there for around 10 seconds. If the RFCOMM channel *does* establish a connection during this time, the router service will remain in the foreground. However, if the RFCOMM channel *does not* establish a connection, the service will exit from the foreground after the timeout has been reached.
  2. **The router service has seen this hardware before, but has never connected:** In this case router service will immediately exit from the foreground after being started.
  3. **The router service has successfully connected to this hardware before:** The router service will remain in foreground for around 20 seconds. If the RFCOMM channel *does* establish a connection during this time, the router service will remain in the foreground. However, if the RFCOMM channel *does not* establish a connection, the service will exit from the foreground after the timeout has been reached.

- If a connection is not established during any of the timeout periods and the service exits the foreground, the service does not typically close immediately. It stays in the background until the Android operating system closes it. This timing is not guaranteed or consistent between devices, and therefore the service should be considered "closed" after this timeout occurs.
- It is possible for the router service to be started after the timeout and close from initial bluetooth connection, if an app is launched with the correct integration it should call a method that potentially restarts the router service (similar to `queryForConnectedService(context)`). However, the hardware will have to perform an SDP query to know that this has occurred.

NUMBER OF RFCOMM CHANNELS

- In most cases, the first SDP query will only return a single RFCOMM channel for the SDL UUID; this is the router service's RFCOMM channel. After connecting to that RFCOMM channel it is recommended to do another SDP query. This is due to some apps not trusting the previously connected router service, and they will start their

own Bluetooth transport with a different RFCOMM channel but still use the same SDL UUID. The head unit should connect to each additional RFCOMM channel with the SDL UUID.
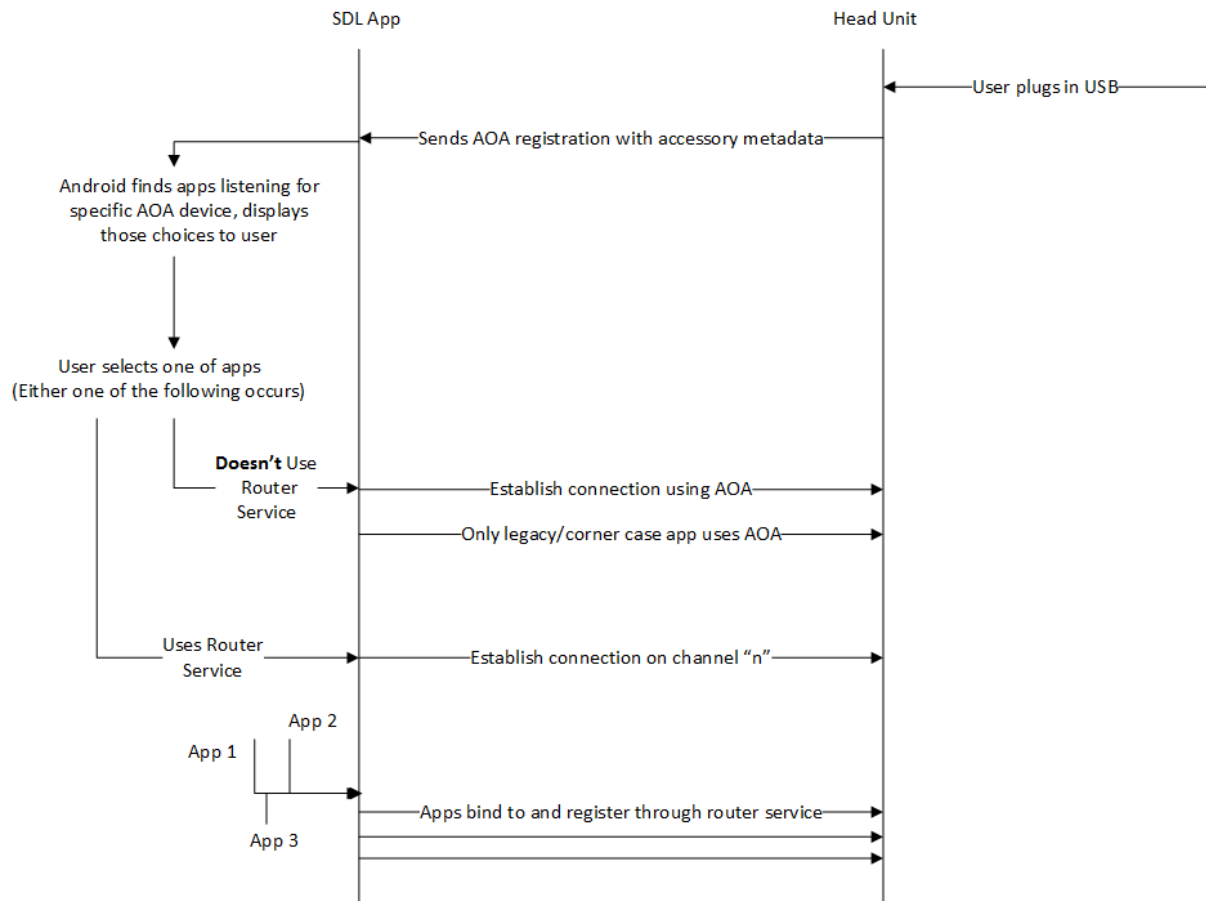
- Due to the previous bullet or other situations (such as if an app crashes), it is also recommend an SDP query be triggered at some other point. Common triggers are user requested button presses, transport disconnection, or app unregistration.
- There is not guarantee on which RFCOMM channel with the SDL UUID is the router service. If during the first SDP query multiple RFCOMM channels are found with the SDL UUID, each one of them will need to be connected in order to ensure that the router service is connected and all other apps connect as well.

USB

Primary or secondary transport

Android has a specification called AOA that allows apps to communicate over a USB connection. The default implementation of AOA on the Android device is very limited. Users must select an app to receive the "USB device" upon connection on the device. However, the SDL Android library leverages its already existing functionality of multiplexing a single transport connection through a router service to allow multiple apps to use the AOA connection.

Diagram

```
          SDL App                                    Head Unit

                                                          ◄──────User plugs in USB──────
              ◄────────Sends AOA registration with accessory metadata────────
              │
              ▼
  Android finds apps listening for
  specific AOA device, displays
     those choices to user
              │
              ▼
  User selects one of apps
  (Either one of the following occurs)

           Doesn't Use
             Router ────────►────────Establish connection using AOA────────────►
             Service
                   ────────────Only legacy/corner case app uses AOA────────────►

           Uses Router ──────►────────Establish connection on channel "n"────────►
             Service
                  App 2
           App 1  │  │
                  │  │──────►────Apps bind to and register through router service────►
              App 3 ──────────────────────────────────────────────────────────────►
                   ──────────────────────────────────────────────────────────────►
```

LIMITATIONS

- AOA has been part of the SDL Android library for a while, however, the ability to multiplex the single AOA connection has only been part of the library since SDL Android Library version 4.7.0 and newer. This means that some apps might still have the older integration that does not support AOA through the router service.
- The library is designed to handle this case as long as the older AOA app updates their library and either updates its integration to support the router service or another app has the newer library that supports the router service.
- The AOA connection is only given to an app that the user selects on the device at the time of connecting the USB cable. This means that the user has to select one of the apps that can support multiplexing the AOA connection for other apps to connect. If a user selects an app that *doesn't* support multiplexing over AOA, only that app will register.

TCP

Secondary transport only

Due to limitations for the iOS platform (discussed in the next section), Android can only use the TCP transport as a secondary transport in production. The hardware will send the connection information after a primary transport is connected through an SDL packet.

iOS

___

BLUETOOTH AND USB

Primary transport only

Apple has a proprietary technology called iAP that apps connect through to communicate data over Bluetooth or USB. Unfortunately, this means that none of that information can be shared here. You will have to build and certify your hardware as part of Apple's MFi program in order to enable the Bluetooth and USB transports, which are also the only primary transport available. Therefore, without MFi certification and building support for the iAP transport, your hardware cannot support production iOS devices and apps.

SINGLE-SESSION VS. MULTI-SESSION CONNECTION SCHEMES

The way IAP works is that you have a protocol string that a hardware device and an app declare that they can use. When iOS sees a hardware device with a known protocol string, it notifies all apps that declare that protocol string. The app can then open a 1-1 connection with the hardware using that protocol string as an identifier for the connection.
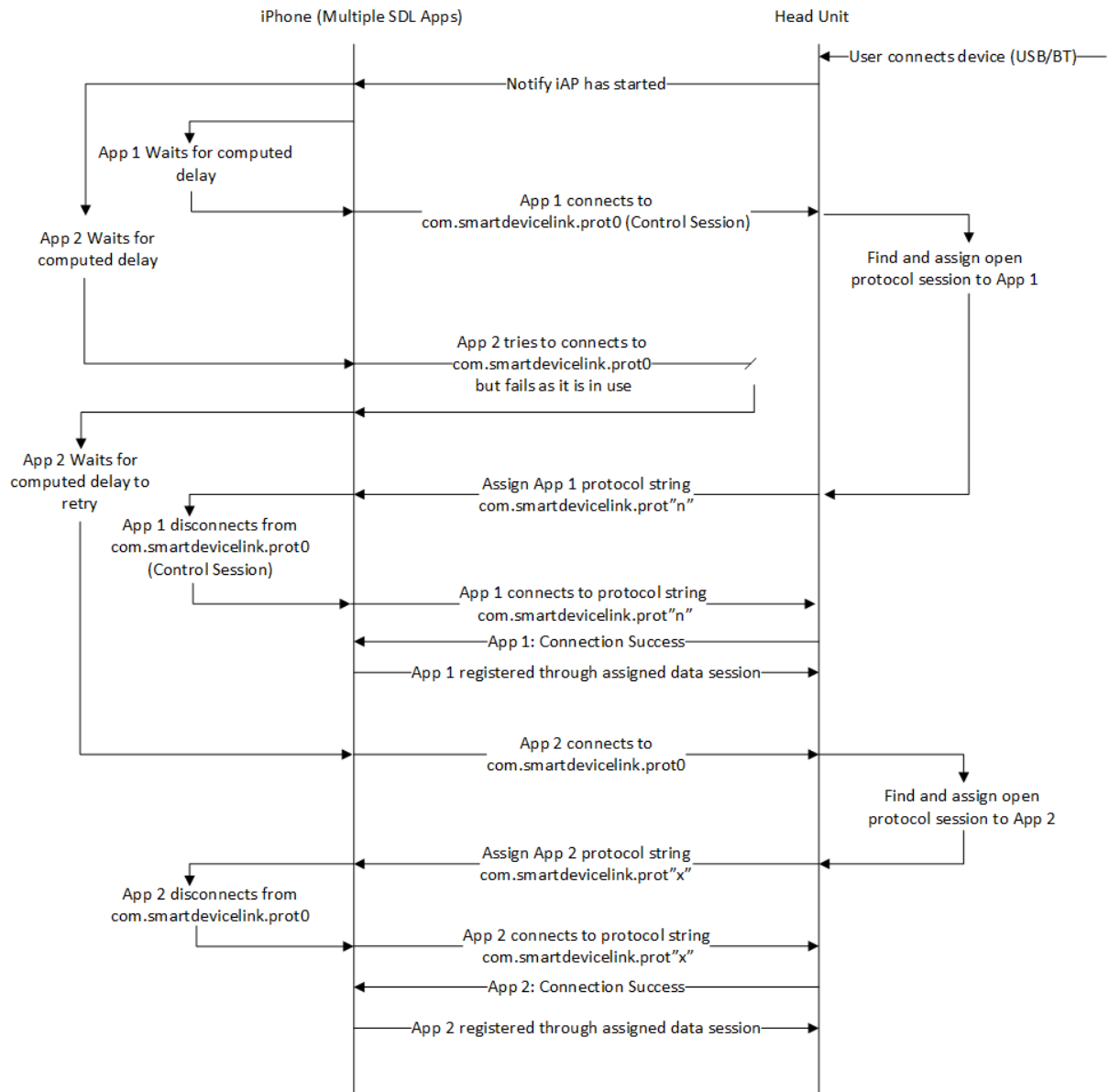
Single-Session Connections (Legacy)

There are two different connection strategies that the iOS library uses. The first is the "hub" strategy for connecting to a single-session iAP protocol. This method is only included in the library as legacy support and should not be implemented in new modules. The hub strategy was developed to allow multiple apps to connect to the hardware before multiple apps could use the same protocol string.

The hardware has to make 30 protocol strings available. Each app will attempt to connect to one protocol (`prot0` , called the control protocol) in order to receive one byte of data. That data is a number 1-29. The app will then disconnect from the control protocol and connect to a data protocol corresponding to the number they received (prot1-prot29).

Because many apps are attempting to connect to the control protocol in order to receive their protocol string number, the app is given a timeout based on a hash of the app's name. The apps will try to connect to the control string after that timeout. If they fail, they'll wait the timeout 3 more times and attempt to connect to get their data protocol number.
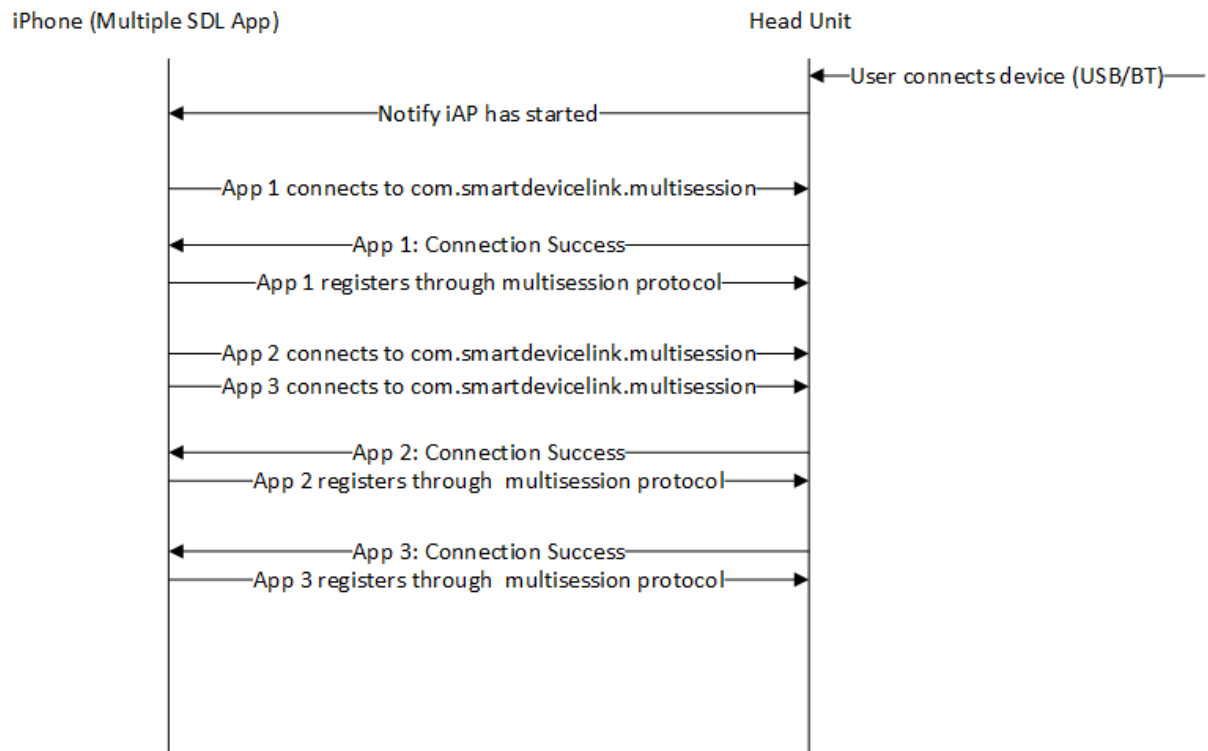
iPhone (1 SDL App)   Head Unit

←—————————————————————————— User connects device (USB/BT) ——————

←——————————————— Notify iAP has started ———————————————

Waits for computed delay

—————— Connects to com.smartdevicelink.prot0 (Control Session) →

Find and assign open protocol session

←—————— Assign protocol string com.smartdevicelink.prot"n" ——————←

Disconnect from
com.smartdevicelink.prot0
(Control Session)

—————— Connects to protocol string com.smartdevicelink.prot"n" →

←————————————————— Connection Success —————————————————

————————————— App registered through assigned data session —————————→

This strategy works, but has some large downsides. The more apps that try to connect to the control protocol, the more collisions will result, and the less reliable the connection becomes. Once more than four or five apps connect, the connection will degrade quickly and some apps may connect sometimes and not other times.

iPhone (Multiple SDL Apps)   Head Unit

                                                                        ◄──User connects device (USB/BT)──
                            ◄────────────Notify iAP has started─────────────

        ┌──────┐
  App 1 Waits for computed
         delay

                            App 1 connects to
                    com.smartdevicelink.prot0 (Control Session)────────────►
  App 2 Waits for
  computed delay   Find and assign open
                                                                    protocol session to App 1

                            App 2 tries to connects to
                    ──com.smartdevicelink.prot0────────────────────╱
                            but fails as it is in use
                            ◄───────────────────────────────────────

  App 2 Waits for
  computed delay to   Assign App 1 protocol string
     retry   ◄────────com.smartdevicelink.prot"n"──────────────────
  App 1 disconnects from
  com.smartdevicelink.prot0
     (Control Session)

                            App 1 connects to protocol string
                    ────────com.smartdevicelink.prot"n"───────────────────►
                    ◄──────────App 1: Connection Success─────────
                    ────App 1 registered through assigned data session────►

                            App 2 connects to
                    ────────com.smartdevicelink.prot0──────────────────────►
                                                                    Find and assign open
                                                                    protocol session to App 2

                            Assign App 2 protocol string
                    ◄────────com.smartdevicelink.prot"x"───────────────────
  App 2 disconnects from
  com.smartdevicelink.prot0
                            App 2 connects to protocol string
                    ────────com.smartdevicelink.prot"x"───────────────────►
                    ◄──────────App 2: Connection Success─────────
                    ────App 2 registered through assigned data session────►

MULTI-SESSION CONNECTIONS (MODERN)

Multi-Session is a newer iAP feature that allows multiple connections to a single protocol
string. Every app can connect to the multi-session protocol string without needing to wait
or needing to manage connection timing. This is the preferred method to connect iOS SDL
apps because it eliminates a lot of the stability issues that have been observed with the
single-session method. See this evolution proposal for its acceptance into the project.

Sequence diagram: iPhone (Multiple SDL App) — Head Unit

- User connects device (USB/BT)
- Notify iAP has started
- App 1 connects to com.smartdevicelink.multisession
- App 1: Connection Success
- App 1 registers through multisession protocol
- App 2 connects to com.smartdevicelink.multisession
- App 3 connects to com.smartdevicelink.multisession
- App 2: Connection Success
- App 2 registers through multisession protocol
- App 3: Connection Success
- App 3 registers through multisession protocol

TCP

Secondary transport only

iOS has a very strict limitation that no TCP connection can be maintained while an app is in the background. For this reason TCP can only be a secondary transport and should only be used for video streaming applications, which already have this limitation. The hardware will send the connection information to the library after a primary transport is connected through an SDL packet, and the library will then take care of attempting to connect to the IP address and port.

Java SE

WEBSOCKET SERVER

Primary transport only

The only transport currently available by default in the SDL Java SE library is a web socket server connection. The Java SE app should usually run directly on the same device as the SDL Core and HMI (i.e. it should be an embedded application).

Determining when the Java SE app should start listening to that web socket port will need to be defined by the SDL Core integrator. This will likely be during start up of the device. The Java SE app can simply be started and wait for the user to select the application from the application list. See the 0158 - Cloud App Transport Adapter proposal for more information.
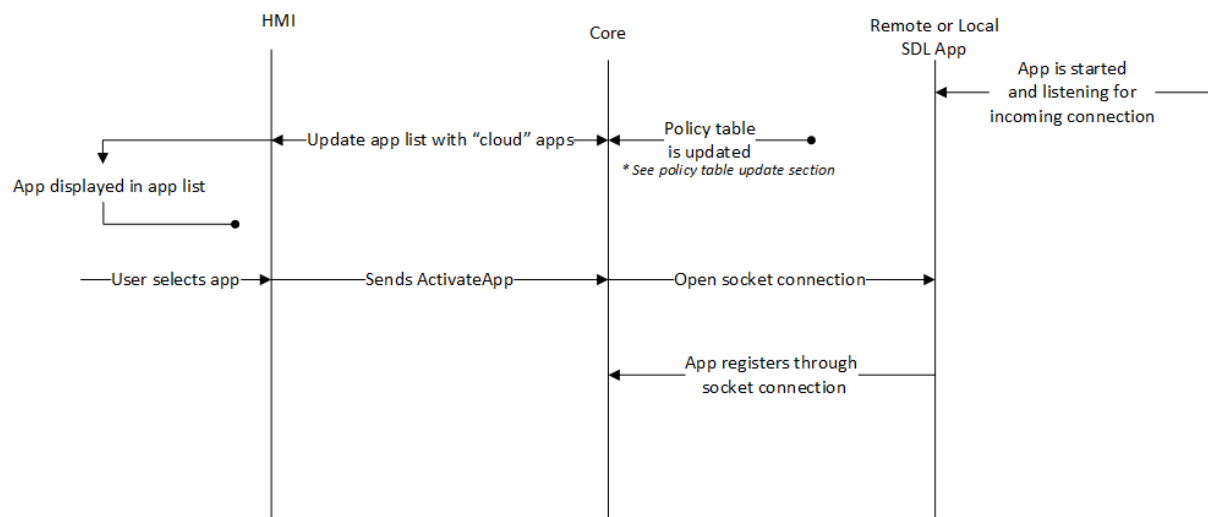
SECONDARY TRANSPORT

Currently the Java SE client library does not support a secondary transport because the WebSocket transport is already high bandwidth.

Java EE

JAVA BEANS (CUSTOM TRANSPORT)

Primary Transport Only

Java EE is usually built around Java Beans in order to easily handle many simultaneous connections that each have their own, independent state. This is another example of a proprietary or separately licensed transport. Because of this, the Java EE library has what's called a Custom Transport. This also developers to hook into the Java EE Bean framework without the SDL library having to include that code into the library itself. Java EE apps are designed to be long running applications that live on an off-board server. The SDL Core implementation will connect out to these applications once they are selected from the app list on the HMI. See the 0158 - Cloud App Transport Adapter proposal for more information.

The previous flow for JavaSE can be followed as the difference will depend on the type of custom transport used. For most cases, using Java Beans is the best flow and will only differ in adding multiple connections to the app via independent instances created by a type of load balancer.

SECONDARY TRANSPORT

Currently the Java EE client library does not support a secondary transport as the websocket transport is already high bandwidth.

Starting an SDL Session

The Protocol Spec defines the lowest SDL layer, and one of the first items that is required to connect an app is start an SDL Session. This includes starting the base RPC and Hybrid services – which are types of data sent using SDL that can be turned on and off – using the Control service which is always available.

The first packet sent is from the app library to the SDL Core implementation. This is a `StartService` control frame message. This packet is very basic and contains only a small amount of information about the app that is connecting and is intended to start the protocol session that all other SDL layers are built on top of. See the protocol spec for more information.

The Core implementation should respond with a `StartServiceACK` that contains specific information as described in the protocol spec. After this moment, the SDL session has been defined and started for an app.
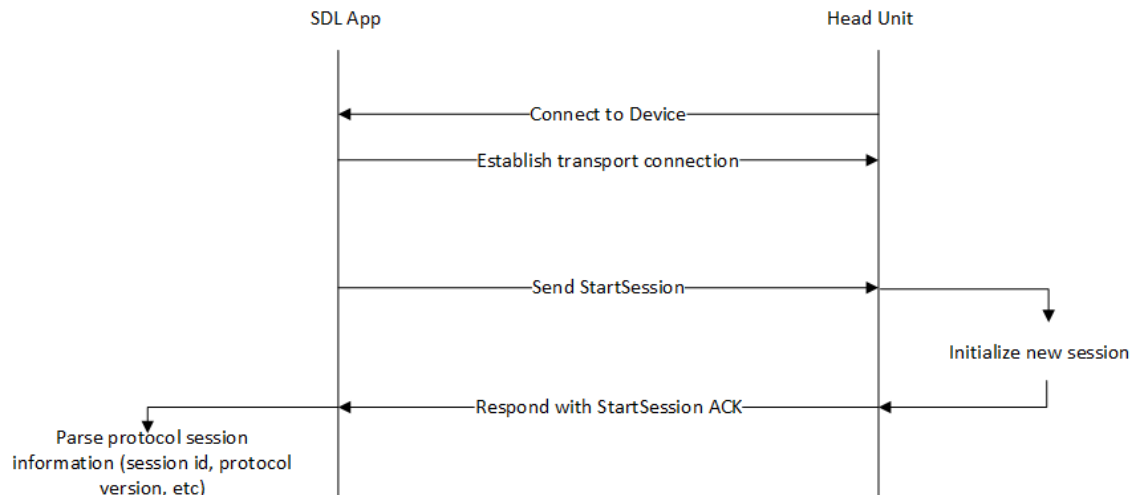
When is Streaming Allowed

SDL Core maintains which apps can stream audio or video via the `audioStreamingState` and `videoStreamingState` parameters of `OnHMIStatus`. For an app to stream it must be in `STREAMABLE` `videoStreamingState` or `AUDIBLE` / `ATTENUATED` `audioStreamingState`. SDL Core knows whether the app can use video services or not depending on the `AppHMIType`. Currently, only `NAVIGATION` and `PROJECTION` apps can use video services. Assuming there is at most one app that can do video streaming at any time, the following table shows how SDL Core shall act regarding `videoStreamingState` in different cases:

CASE	CURRENT VIDEO SOURCE	SDL APP2 COMES TO FULL	RESULTS	ONHMISTATUS VIDEOSTREAMING STATE TO APP1	ONHMISTATUS VIDEOSTREAMINGSTATE TO APP2
1	NONE	app2 does not use video	no app is the video source	NA	NOT_STREAMABLE
2	NONE	app2 uses video	app2 is the video source	NA	STREAMABLE
3	app1	app2 does not use video	app1 is **still** the video source	No Change	NOT_STREAMABLE
4	app1	app2 uses video	app2 is the **new** video source	NOT_STREAMABLE	STREAMABLE
5	app1	CarPlay/Android Auto is the active screen	no SDL app is the video source	NOT_STREAMABLE	NA

> 📝 **NOTE**
>
> If an app does not stop video service after receiving `onHMIStatus` with `videoStreamingState` = `NOT_STREAMABLE` for some time, SDL Core shall send a stop service control frame to the app.

The bool parameter `isMediaApplication` in the register app interface request indicates if an application supports the `AUDIBLE` streaming state. There are two types of projection

applications:

- If `AppHMIType`=`PROJECTION` and `isMediaApplication`=false, we call it a non-media projection application.
- If `AppHMIType`=`PROJECTION` and `isMediaApplication`=true, or `AppHMIType`=`PROJECTION` & `MEDIA`, we call it a media projection application.

We assume that media projection applications stream audio data via either Bluetooth A2DP (for android) or iAP (for iOS) as regular media applications do. We also assume projection applications do not use binary audio services. The following table shows how SDL Core acts regarding `audioStreamingState` in different cases.

CASE	CURRENT AUDIO SOURCE (APPHMITYPE)	SDL APP2 ACTIVATED OR SYSTEM EVENT	APP2 APPHMITYPE	APP2 IS MEDIA	RESULTS	ONHMISTATUS AUDIOSTREAMINGSTATE TO APP1	ONHMISTATUS AUDIOSTREAMINGSTATE TO APP2
1	none	app2 does not use audio	any	false	no app is the audio source	NA	NOT_AUDIBLE
2.1	none	app2 uses audio	NAVIGATION	false	app2 is the audio source	NA	AUDIBLE
2.2	none	app2 uses audio	any	true	app2 is the audio source	NA	AUDIBLE
3	app1 (any)	app2 does not use audio	any	false	app1 is the audio source	No Change	NOT_AUDIBLE
4.1	app1 (NAVIGATION or COMMUNICATION)	app2 uses audio	same as app1	false	app2 is the audio source	NOT_AUDIBLE	AUDIBLE

CASE	CURRENT AUDIO SOURCE (APPHMITYPE)	SDL APP2 ACTIVATED OR SYSTEM EVENT	APP2 APPHMITYPE	APP2 IS MEDIA	RESULTS	ONHMISTATUS AUDIOSTREAMINGSTATE TO APP1	ONHMISTATUS AUDIOSTREAMINGSTATE TO APP2
4.2	app1 (NAVIGATION or COMMUNICATION)	app2 uses audio	others	true	app1 and app2 are the audio sources	AUDIBLE	AUDIBLE / ATTENTUATED or NOT_AUDIBLE *
4.3	app1 (Non-NAVIGATION, NON-COMMUNICATION)	app2 uses audio	NAVIGATION or COMMUNICATION	false	app1 and app2 are the audio sources	AUDIBLE / ATTENTUATED or NOT_AUDIBLE *	AUDIBLE
4.4	app1 (Non-NAVIGATION, NON-COMMUNICATION)	app2 uses audio	others	true	app2 is the audio source	NOT_AUDIBLE	AUDIBLE

CASE	CURRENT AUDIO SOURCE (APP HMITYPE)	SDL APP2 ACTIVATED OR SYSTEM EVENT	APP2 APPHMITYPE	APP2 IS MEDIA	RESULTS	ONHMISTATUS AUDIOSTREAMINGSTATE TO APP1	ONHMISTATUS AUDIOSTREAMINGSTATE TO APP2
5.1	app1 (any)	PHONE_CALL / EMERGENCY_EVENT / AUDIO_SOURCE / EMBEDDED_NAVI	NA	NA	app1 is not the audio source, system set the audio source	NOT_AUDIBLE	NA
5.2	app1 (any)	TTS Start	NA	NA	depending on mixing audio support, app1 can be audio source	NOT_AUDIBLE / ATTENUATED	NA
5.3	app1 (any)	CarPlay /Android Auto is the active screen (DEACTIVATE_HMI)	NA	NA	no SDL app is the video source	NOT_AUDIBLE	NA

*In the case of co-existence of a NAVIGATION app and a MEDIA or COMMUNICATION app, when the NAVIGATION app does not start audio steaming service, the MEDIA/COMMUNICATION app is AUDIBLE ; When the NAVIGATION app starts audio streaming service, the MEDIA/COMMUNICATION app is either ATTENUATED if the system supports mixing or NOT_AUDIBLE if the system does not support mixing.

> ### NOTE
>
> The transition of videoStreamingState and audioStreamingState is independent of the transition of hmiLevel . However, the transition of hmiLevel depends on both audioStreamingState and videoStreamingState . SDL Core moves a LIMITED level media/projection/navigation app which is NOT_AUDIBLE and NOT_STREAMABLE to BACKGROUND HMI level. There are at most two media/projection/navigation apps which can be placed in HMI level LIMITED . At HMI level LIMITED , an app can be either AUDIBLE (including ATTENTUATED) or STREAMABLE or both.

Registering an Application

After the app starts the SDL session, it's time to send more information to the SDL Core implementation about the application that is connecting. The library will send a RegisterAppInterface RPC request to Core. What information can be sent is found here. Core passes most of this information to the HMI so that the HMI can present this application on an app list for the user to start.

After Core and the HMI have accepted the app, Core will create a RegisterAppInterface response to send back to the app. This will contain a lot of information about the system that the app library will need to know in order to adjust the app's UI and adapt to this specific system. The supplied parameters can be found in the RPC Spec.
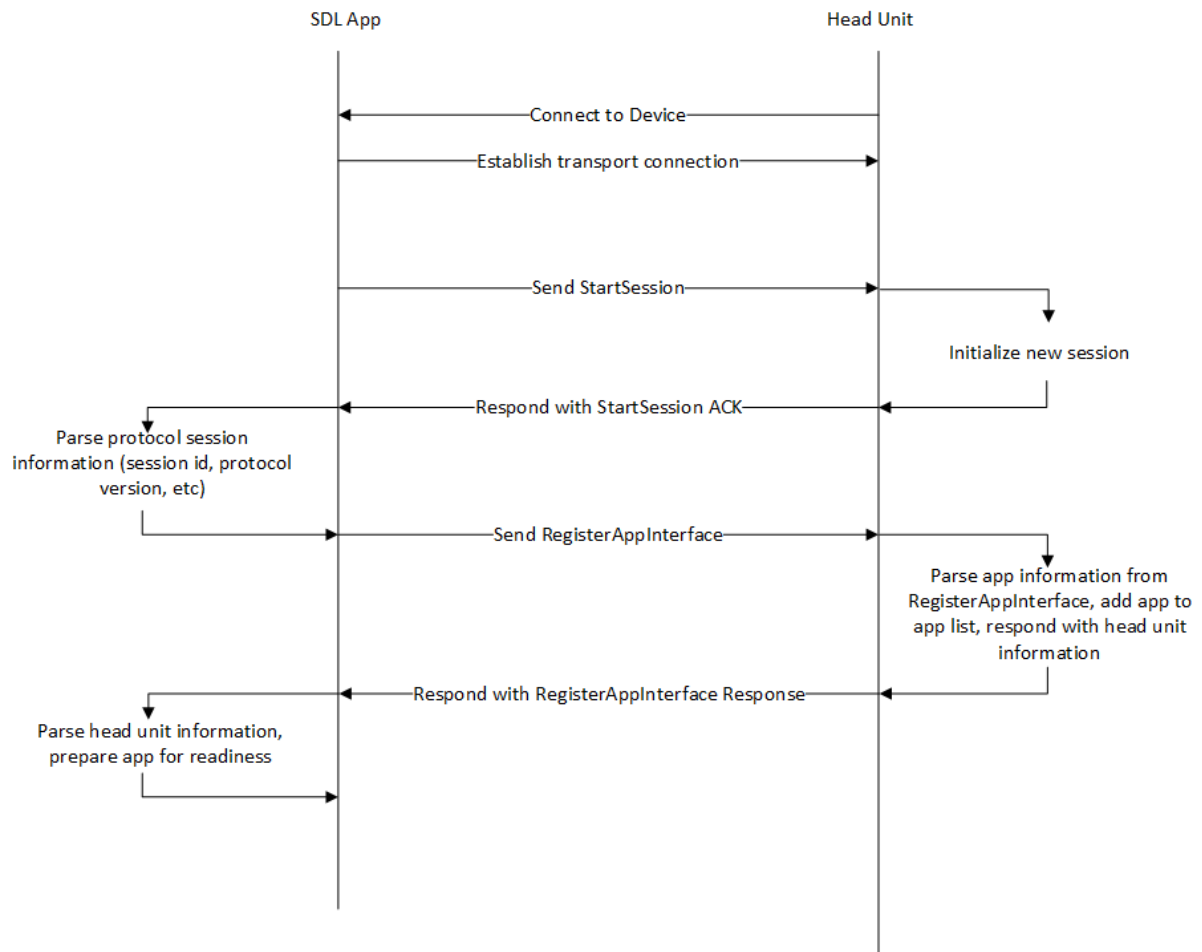
Once the app is registered, Core usually sends a few more notifications to the app. These include an `OnHMIStatus`, `OnPermissionsChange`, `OnHashChange`, and `OnDriverDistraction`.

Post-Registration Notifications

OnHMIStatus

This is a required notification from Core to be sent to the app. This sets the basic HMI status for the app on the head unit. At this point the app knows that it registered and displayed on an app list waiting for a user to select it. You can find more information about OnHMIStatus parameters in the RPC spec.

OnHashChange

If the implementation of Core supports hash resumption, any time an item that can be restored with the correct hash changes this notification will be sent. Since the app just registered with information that updated the app record on Core, this notification is sent to the app. You can find more information about OnHashChange parameters in the RPC spec.

OnPermissionsChange

After the app has registered, Core will check the policy table to see if the app has any permission updates. You can find more information about OnPermissionsChange parameters in the RPC spec.

OnDriverDistraction

This is a required notification from Core to be sent to the app. It defines information about whether or not the driver is currently in a distracted state (based on factors like the vehicle speed). You can find more information about OnDriverDistraction parameters in the RPC spec.

After Registration

After the `OnHMIStatus` is received, the app will likely send a few RPCs that help get the head unit ready to display that application. The most common RPCs will be to set an app icon and to receive the list of files the application has stored on the head unit. The app icon is set up with a `PutFile` followed by a `SetAppIcon` request, while the list of files is retrieved with a `ListFiles` request.

Some apps will send more RPCs at this time, but based on the policy table they might be limited to a certain number.

SDL App   Head Unit

Connect to Device
Establish transport connection

Send StartSession

Initialize new session

Respond with StartSession ACK

Parse protocol session
information (session id, protocol
version, etc)

Send RegisterAppInterface

Parse app information from
RegisterAppInterface, add app to
app list, respond with head unit
information

Respond with RegisterAppInterface Response

Parse head unit information,
prepare app for readiness

OnHMIStatus: HMI_NONE
OnHashChange: (updated hashID)
onPermissionsChange: (updated permisions)

Send PutFile for App Icon
Respond with PutFile Response
Send SetAppIcon
Respond with SetAppIcon Response

Activating the App

At this point, the app should have started its SDL protocol session, registered through the RPC service, and be displayed in some sort of app list. The next step is to activate the app once it should be displayed on the display. This will inform the app that their HMI status has changed to one that is visible, and the app will start to send its content to the head unit.

This content will include setting their layout template, and sending default text and images to display. They also have the opportunity to build out their menus and submenus. If the head unit supports voice recognition, the app will send over its VR synonyms for voice commands.

```
                        SDL App                                    Head Unit

                           │←──────────── Connect to Device ──────────│
                           │                                          │
                           │──────── Establish transport connection ──→│
                           │                                          │
                           │                                          │
                           │──────────── Send StartSession ───────────│
                           │                                          │↓
                           │                                          │
                           │                                    Initialize new session
                           │                                          │
                           │←─────── Respond with StartSession ACK ────│←┘
                           │↓
             Parse protocol session
          information (session id, protocol
                  version, etc)
                           │
                           └→│────────── Send RegisterAppInterface ─────│
                             │                                          │↓
                             │                                  Parse app information from
                             │                                  RegisterAppInterface, add app to
                             │                                  app list, respond with head unit
                             │                                       information
                             │                                          │
                             │←──── Respond with RegisterAppInterface Response ──│←┘
                             │↓
             Parse head unit information,
             prepare app for readiness
                           │
                           └→│
                             │
                             │←──────── OnHMIStatus: HMI_NONE ──────────│
                             │←──────── OnHashChange: (updated hashID) ─│
                             │←──── onPermissionsChange: (updated permisions) ──│
                             │                                          │
                             │──────── Send PutFile for App Icon ───────│
                             │←──────── Respond with PutFile Response ───│
                             │────────────── Send SetAppIcon ───────────│
                             │←──────── Respond with SetAppIcon Response ─│
                             │                                          │
                             │                                    User selects app
                             │                                     from app list
                             │                                          │
                             │←──────── OnHMIStatus: HMI_FULL ──────────│←┘
                             │↓
             Prepare content to be sent to
                    head unit HMI
                           │
                           └→│─────── Sends Show with text updates ──────│
                             │                                          │↓
                             │                                      Update HMI
                             │←──────── Respond with Show Response ──────│←┘
                             │─────── Send PutFile for Primary Graphic ───│
                             │←──────── Respond with PutFile Response ────│
                             │─────── Sends Show with image updates ──────│
                             │                                          │↓
                             │                                      Update HMI
                             │←──────── Respond with Show Response ──────│←┘
                             │                                          │
```

SmartDeviceLink Protocol

Current Version: 5.4.1

1. Overview

The SmartDeviceLink protocol specification describes the method for establishing communication between an application and head unit and registering the application for continued communication with the head unit. The protocol is used as the base formation of packets sent from one module to another.

All new SDL implementations should implement the newest version of the protocol.

1.1 Common Terms

TERM	DESCRIPTION
Module / Head Unit	Hardware implementing the sdl_core software
Application	Smart device application that implements the proxy library (iOS or Android)

2. Frames

All transported data is formed with a header followed by an optional payload. The combination of header and payload is referred to as a frame.

2.1 Version 1 Frame Header

Deprecated: Protocol versions 2 and higher. Only used as initial StartService
packet for establishing communication and version negotiation from
application

Byte 1			Byte 2	Byte 3	Byte 4
Version	C	Frame Type	Service Type	Frame Info	Session ID

Byte 5	Byte 6	Byte 7	Byte 8
Data Size			

2.2 Version 2 Frame Header

Required: Protocol versions 2 and higher

Byte 1			Byte 2	Byte 3	Byte 4
Version	E	Frame Type	Service Type	Frame Info	Session ID

Byte 5				Byte 6	Byte 7	Byte 8
Data Size						

Byte 9				Byte 10	Byte 11	Byte 12
Message ID						

2.3 Frame Header Fields

Field	Size	Description
Version	4 bit	**Protocol Version** 0x1 Protocol version 1 - uses a version 1 Frame Header 0x2 Protocol version 2 - uses a version 2 Frame Header 0x3 Protocol version 3 - uses a version 2 Frame Header 0x4 Protocol version 4 - uses a version 2 Frame Header 0x5 Protocol version 5 - uses a version 2 Frame Header 0x6 - 0xF Reserved
C	1 bit	**Compression Flag** 0x0 This packet is not compressed 0x1 This packet is compressed **Note:** Only available in Protocol Version 1
E	1 bit	**Encryption Flag** 0x0 This packet is not encrypted 0x1 This packet is encrypted **Note:** Only available in Protocol Version 2 and higher. Must be always set to zero for a First Frame
Frame Type	3 bit	0x00 Control Frame 0x01 Single Frame 0x02 First Frame 0x03 Consecutive Frame 0x04 - 0x07 Reserved
Service Type	8 bit	0x00 Control Service 0x01 - 0x06 Reserved 0x07 Remote Procedure Call (RPC) Service 0x08 - 0x09 Reserved 0x0A Audio Service 0x0B Video Service 0x0C - 0x0E Reserved 0x0F Bulk Data (Hybrid Service) 0x10 - 0xFF Reserved
Fram	8 bit	**Frame Type = 0x00 (Control Frame)**

e Info		0x00 Heartbeat
		0x01 Start Service
		0x02 Start Service ACK
		0x03 Start Service NAK
		0x04 End Service
		0x05 End Service ACK
		0x06 End Service NAK
		0x07 Register Secondary Transport
		0x08 Register Secondary Transport ACK
		0x09 Register Secondary Transport NAK
		0x0A - 0xFC Reserved
		0xFD Transport Event Update
		0xFE Service Data ACK
		0xFF Heartbeat ACK
		Frame Type = 0x01 (Single Frame)
		0x00 - 0xFF Reserved
		Frame Type = 0x02 (First Frame)
		0x00 - 0xFF Reserved
		Frame Type = 0x03 (Consecutive Frame)
		0x00 Last Frame
		0x01 - 0xFF Frame Number
Session ID	8 bit	The session identifier
Data Size	32 bit	**Frame Type = 0x00 (Control Frame)** 0x0 - 0xFFFFFFFF reserved. **Frame Type = 0x02 (First Frame)** 0x08 The data size for a first frame is always 8 bytes. In the payload, the first four bytes denote the Total Size of the data contained in all consecutive frames. This is always the size of whole non-encrypted payload (even if consecutive frames are encrypted). The second four bytes denote the number of consecutive frames following this one **Frame Type = 0x01 or 0x03 (Single or Consecutive Frame)** The total bytes in this frame's payload. If frame is encrypted this is the size of encrypted payload, otherwise size of non-encrypted payload.
Message ID	32 bit	The message identifier, used to uniquely identify this message. **Note:** Only included in protocol version 2 frame headers and higher

2.4 Max Transport Units

The max transport unit (MTU) of a frame varies based on version. The MTU includes the frame header and payload. The current supported versions and their MTU's respectively are described below.

VERSION	MTU (BYTES)
1	1500
2	1500
3	131,084
4	131,084
5	131,084 default or negotiated *(See Control Frame Payloads)*

2.4.1 PAYLOAD SIZE

The payload size is determined by the MTU - Frame Header Size.

VERSION	MAX PAYLOAD SIZE (BYTES)
1	1488
2	1488
3	131,072
4	131,072
5	131,072 default or (Negotiated MTU - 12 bytes) *(See Control Frame Payloads)*

2.4.2 ENCRYPTED MTU

While the supported MTU is the maximum size for that version, if a frame is encrypted it will be subject to the MTU of that encryption protocol as well. That means the MTU will have to be the minimum between SDL's MTU and the encryption protocol's MTU.

3. Frame Types

3.1 Control Frame

Control frames are the lowest-level type of packets. They can be sent over any of the defined services. They are used for the control of the services in which they are sent.

3.1.1 SPECIAL HEADER DEFINITIONS:

HEADER VALUE	EXPECTED VALUES	DESCRIPTION
Frame Info	0x00 - 0x06, 0xFE, 0xFF	See below "Frame Info Definitions"
Data Size	0x00, 0x04	0x00 - Majority of control packets do not have payloads 0x04 - Used for StartServiceACK where the payload is a HashID

3.1.2 FRAME INFO DEFINITIONS:

FRAME INFO VALUE	NAME	DESCRIPTION
0x00	Heartbeat	A ping packet that is sent to ensure the connection is still active and the service is still valid
0x01	Start Service	Requests that a specific type of service is started
0x02	Start Service ACK	Acknowledges that the specific service has been started successfully
0x03	Start Service NAK	Negatively acknowledges that the specific service was *not* started
0x04	End Service	Requests that a specific type of service is ended
0x05	End Service ACK	Acknowledges that the specific service has been ended successfully
0x06	End Service NAK	Negatively acknowledges that the specific service was *not* ended or has not yet been started
0x07	Register Secondary Transport	Request for a session registered on a primary transport to use a secondary transport. This frame should only be sent on the Secondary Transport that the session is requesting.

FRAME INFO VALUE	NAME	DESCRIPTION
0x08	Register Secondary Transport ACK	Acknowledges that the supplied session is registered to use the requested Secondary Transport. The application is only allowed to send additional frames on the Secondary Transport after this frame is received. This frame must be sent on the Secondary Transport in which the original request was sent.
0x09	Register Secondary Transport NAK	Negatively acknowledges that the session is not registered or able to use the current Secondary Transport. The application cannot use this transport for any other messages. This frame must be sent on the Secondary Transport in which the original request was sent.
0xFD	Transport Event Update	Indicates that status or configuration of one or more transports are updated. This frame must only be sent after the successful starting of the RPC service which includes the protocol version negotiation.
0xFE	Service Data ACK	*Deprecated*

FRAME INFO VALUE	NAME	DESCRIPTION
0xFF	Heartbeat ACK	Acknowledges that a Heartbeat control packet has been received

---

3.1.3 PAYLOADS

Added: Protocol Version 5

Note: All parameters are optional

Control frames use BSON to store payload data. All payload types are directly from the BSON spec. Each control frame info type will have a defined set of available data. Most types will also have differently available data based on their service type.

If there is no data to send for a given parameter, the parameter should not be included.

Note: Heartbeat, Heartbeat ACK, and Service Data ACK control frame types are not covered for any service as they were deprecated before payloads were introduced.

3.1.3.1 CONTROL SERVICE

3.1.3.1.1 Register Secondary Transport

No parameters

3.1.3.1.2 Register Secondary Transport ACK

No parameters

3.1.3.1.3 Register Secondary Transport NAK

TAG NAME	TYPE	INTRODUCED	DESCRIPTION
reason	String	5.1.0	A string describing the reason of failure

3.1.3.1.4 Transport Event Update

TAG NAME	TYPE	INTRODUCED	DESCRIPTION
tcpIpAddress	String	5.1.0	IP address that can be used to establish a TCP connection. It can be IPv4 address (example: "192.168.1.1") or IPv6 address (example: "fd12:3456:789a::1"). An empty string indicates that the TCP transport becomes unavailable.
tcpPort	int32	5.1.0	TCP Port number that can be used along with the supplied tcpIpAddress to establish a TCP connection. If parameter is included, the tcpIpAddress parameter must also be included.

3.1.3.2 RPC SERVICE

3.1.3.2.1 Start Service

Note: While this includes a payload, it will remain a v1 frame header to ensure backwards compatibility with older systems.

TAG NAME	TYPE	INTRODUCED	DESCRIPTION
protocolVersion	String	5.0.0	The max version of the protocol supported by client requesting service to start. Must be in the format *"Major.Minor.Patch"*

3.1.3.2.2 Start Service ACK

TAG NAME	TYPE	INTRODUCED	DESCRIPTION
protocolVersion	String	5.0.0	The negotiated version of the protocol. Must be in the format *"Major.Minor.Patch"*. The frame header version should match the major version exactly.
hashId	int32	5.0.0	Hash ID to identify this session and used when sending an EndService control frame for the RPC service type
mtu	int64	5.0.0	Max transport unit to be used for this service

TAG NAME	TYPE	INTRODUCED	DESCRIPTION
secondaryTransports	String Array	5.1.0	Array of transport types which are allowed to be used as a Secondary Transport. Refer to the table below for possible values. As of this protocol spec version (5.1.0) only a single Secondary Transport may be used beyond a primary transport for a given session. If there are no currently available Secondary Transports or the functionality is not supported, this parameter should be omitted or be an empty array.
TAG NAME	TYPE	INTRODUCED	DESCRIPTION

TAG NAME	TYPE	INTRODUCED	DESCRIPTION
audioServiceTransports	int32 array	5.1.0	Ordered list of transport priority types that support the audio service (0x0A). Only the values of 1 ("Primary Transport") or 2 ("Secondary Transport") shall be used. If both the primary and secondary transport support the audio service, both should be included (1 and 2) in the order they are preferred; otherwise only the single transport priority type should be included. An application must not start the audio service on a transport priority type that is not listed in the array. If this parameter is not included the Primary Transport should be used for the audio service.

TAG NAME	TYPE	INTRODUCED	DESCRIPTION

TAG NAME	TYPE	INTRODUCED	DESCRIPTION
videoServiceTransports	int32 array	5.1.0	Ordered list of transport priority types that support the video service (0x0B). Only the values of 1 ("Primary Transport") or 2 ("Secondary Transport") shall be used. If both the primary and secondary transport support the video service, both should be included (1 and 2) in the order they are preferred; otherwise only the single transport priority type should be included. An application must not start the video service on a transport priority type that is not listed in the array. If this parameter is not included the Primary Transport should be used for the video service.

TAG NAME	TYPE	INTRODUCED	DESCRIPTION

TAG NAME	TYPE	INTRODUCED	DESCRIPTION
authToken	String	5.2.0	Included exclusively when communicating with cloud applications. This token is used by a cloud application to authenticate a user account associated with the vehicle.
make	String	5.4.0	Vehicle make value. Used by OEM exclusive apps to identify whether current vehicle is supported or not.
model	String	5.4.0	Vehicle model value. Used by OEM exclusive apps to identify whether current vehicle is supported or not.
modelYear	String	5.4.0	Vehicle model year value. Used by OEM exclusive apps to identify whether current vehicle is supported or not.

TAG NAME	TYPE	INTRODUCED	DESCRIPTION
trim	String	5.4.0	Vehicle trim value. Used by OEM exclusive apps to identify whether current vehicle is supported or not.
systemSoftwareVersion	String	5.4.0	Vehicle system software version value. Can be specified in any format desired by the OEM.
systemHardwareVersion	String	5.4.0	Vehicle system hardware version value. Can be specified in any format desired by the OEM.

list of transport type strings

STRING	DESCRIPTION
IAP_BLUETOOTH	iAP over Bluetooth
IAP_USB	iAP over USB where it is not possible to distinguish between host or device mode
IAP_USB_HOST_MODE	iAP over USB, and the phone is running as host
IAP_USB_DEVICE_MODE	iAP over USB, and the phone is running as device
IAP_CARPLAY	iAP over Carplay wireless
SPP_BLUETOOTH	Bluetooth SPP.
AOA_USB	Android Open Accessory
TCP_WIFI	TCP connection over Wi-Fi

3.1.3.2.3 Start Service NAK

TAG NAME	TYPE	INTRODUCED	DESCRIPTION
rejectedParams	String Array	5.0.0	An array of rejected parameters
reason	String	5.3.0	A string describing the reason of failure

3.1.3.2.4 End Service

TAG NAME	TYPE	INTRODUCED	DESCRIPTION
hashId	int32	5.0.0	Hash ID supplied in the StartService ACK for this service type

3.1.3.2.5 End Service ACK
3.1.3.2.6 End Service NAK

TAG NAME	TYPE	INTRODUCED	DESCRIPTION
rejectedParams	String Array	5.0.0	An array of rejected parameters such as: [hashId]
reason	String	5.3.0	A string describing the reason of failure

3.1.3.3 AUDIO SERVICE

3.1.3.3.1 Start Service

No parameters

3.1.3.3.2 Start Service ACK

TAG NAME	TYPE	INTRODUCED	DESCRIPTION
mtu	int64	5.0.0	Max transport unit to be used for this service. If not included the client should use the one set via the RPC service or protocol version default.

3.1.3.3.3 Start Service NAK

TAG NAME	TYPE	INTRODUCED	DESCRIPTION
rejectedParams	String Array	5.0.0	An array of rejected parameters such as: [videoProtocol, videoCodec]
reason	String	5.3.0	A string describing the reason of failure

3.1.3.3.4 End Service

No parameters

3.1.3.3.5 End Service ACK

No parameters

3.1.3.3.6 End Service NAK

TAG NAME	TYPE	INTRODUCED	DESCRIPTION
rejectedParams	String Array	5.0.0	An array of rejected parameters such as: [hashId]
reason	String	5.3.0	A string describing the reason of failure

3.1.3.4 VIDEO SERVICE

3.1.3.4.1 Start Service

TAG NAME	TYPE	INTRODUCED	DESCRIPTION
height	int32	5.0.0	Desired height from the client requesting the video service to start
width	int32	5.0.0	Desired width from the client requesting the video service to start
videoProtocol	String	5.0.0	Desired video protocol to be used. See VideoStreamingProtocol RPC
videoCodec	String	5.0.0	Desired video codec to be used. See VideoStreamingCodec RPC

3.1.3.4.2 Start Service ACK

TAG NAME	TYPE	INTRODUCED	DESCRIPTION
mtu	int64	5.0.0	Max transport unit to be used for this service. If not included the client should use the one set via the RPC service or protocol version default.
height	int32	5.0.0	Accepted height from the client requesting the video service to start
width	int32	5.0.0	Accepted width from the client requesting the video service to start
videoProtocol	String	5.0.0	Accepted video protocol to be used. See VideoStreamingProtocol RPC
videoCodec	String	5.0.0	Accepted video codec to be used. See VideoStreamingCodec RPC

3.1.3.4.3 Start Service NAK

TAG NAME	TYPE	INTRODUCED	DESCRIPTION
rejectedParams	String Array	5.0.0	An array of rejected parameters such as: [videoProtocol, videoCodec]
reason	String	5.3.0	A string describing the reason of failure

3.1.3.4.4 End Service

No parameters

3.1.3.4.5 End Service ACK

No parameters

3.1.3.4.6 End Service NAK

TAG NAME	TYPE	INTRODUCED	DESCRIPTION
rejectedParams	String Array	5.0.0	An array of rejected parameters such as: [hashId]
reason	String	5.3.0	A string describing the reason of failure

3.2 Single Frame

A frame of type Single Frame contains all the data for a particular packet in the payload. The majority of frames sent over the protocol utilize this frame type.
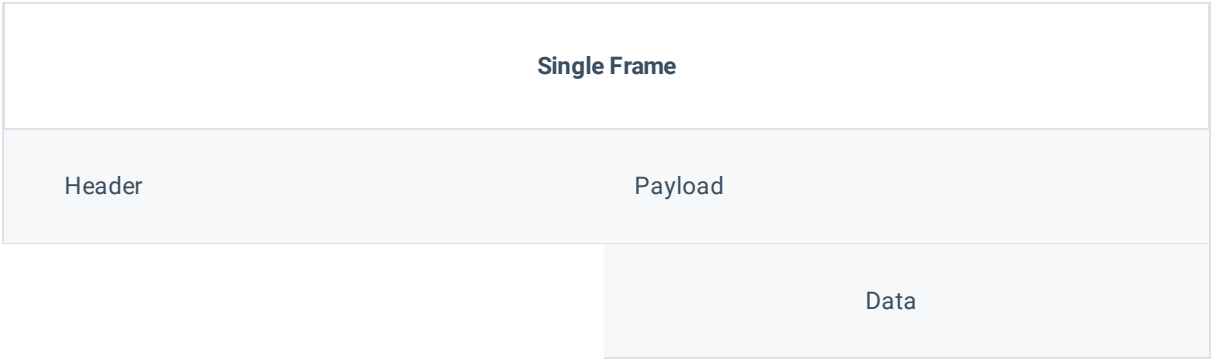
Single Frame	
Header	Payload
	Data

HEADER VALUE	EXPECTED VALUES	DESCRIPTION
Frame Info	0x00	Reserved
Data Size	0x01-0xFFFFFFFF	Total payload size in bytes for this frame

3.3 Multiple Frame Packets

Some payloads will be larger than the maximum transport unit will allow. If that is the case, the payload will be broken up over multiple frames. These frame types are First and Consecutive.

Data				
Data Chunk 1	Data Chunk 2	. . .	Data Chunk n	

First Frame

Payload

Consecutive

Frame 1

Header	Payload

Consecutive Frame 2

Header	Payload

-
-
-

Consecutive Frame

Header Payload

3.3.1 FIRST FRAME

The First Frame in a multiple frame payload contains information about the entire sequence of frames so that the receiving end can correctly parse all the frames and reassemble the entire payload. The payload of this frame is only eight bytes and contains information regarding the rest of the sequence.

3.3.1.1 PAYLOAD:

Byte	0	1	2	3	4	5	6	7
	Total size of the original payload being parsed				Number of Consecutive Frames in this sequence			

3.3.1.2 SPECIAL HEADER DEFINITIONS:

HEADER VALUE	EXPECTED VALUES	DESCRIPTION
Frame Info	0x00	Reserved
Data Size	0x08	This frame contains a fixed data size (8 bytes) for the payload.

3.3.2 CONSECUTIVE FRAME

The Consecutive Frames in a multiple frame payload contain the actual raw data of the original payload. The parsed payload contained in each of the Consecutive Frames' payloads should be buffered until the entire sequence is complete.

3.3.2.1 SPECIAL HEADER DEFINITIONS:

HEADER VALUE	EXPECTED VALUES	DESCRIPTION
Frame Info	0x00 - 0xFF	Values 0x01 - 0xFF are used incrementally as each consecutive frame is created and sent in the sequence. eg The first consecutive packet in the sequence will have the value 0x01, the next consecutive frame that contains the next chunk of data in the sequence will have the value 0x02. If the sequence reaches 0xFF with more frames to create, it shall rollover to 0x01 **not** 0x00 as it is reserved. 0x00 is only used for the last consecutive frame in a multi-frame sequence and the last frame must have this value.
Data Size	0x01 - 0xFFFFFFFF	Payload size in bytes for only this frame

4. Establishing Communication

4.1 Transport Layer

> Required: All Protocol Versions

A physical transport must be established between a head unit and an application before an SDL session can start.

4.2 Version Negotiation

Required: All Protocol Versions

4.2.1 OVERVIEW

Once a physical transport is established, each application must negotiate the maximum supported protocol version with the head unit. To establish basic communication and register with the head unit, the application must start an RPC service (Service Type: 0x07), using a *Version 1 Protocol Header*.

There are two types of version negotiation. Protocol versions 1 through 4 use an old style of negotiation, where as versions 5 and newer use a faster and more intelligent negotiation scheme.

4.2.1.1 VERSION 1-4 NEGOTIATION

Required for Protocol Versions 1 through 4

PROXY	DIRECTION	CORE
StartService **Version:** v1 **Payload:** no payload	————>	
	<————	StartServiceACK **Version:** Max supported by Core **Payload:** raw bytes for hashID
SingleFrame *(or other RPC supporting Frame Type)* **Version:** Highest version supported by both Core and Proxy **Payload:** Lots of bytes	————>	Sets negotiated version.

4.2.1.2 VERSION 5+ NEGOTIATION

Required for Protocol Versions 5 and newer

PROXY	DIRECTION	CORE
StartService **Version:** v1 **Payload:** Constructed payload [protocolVersion: 5.x.x]	————>	**v4 Core**: Ignores payload, sends protocol version 4 frame and uses previous negotiation scheme. **v5+ Core**: Reads in payload data, uses this information to determine version.
	<————	**StartServiceACK** **Version:** Highest version supported by both Core and Proxy **Payload:** Constructed payload [protocolVersion: 5.x.x, hashId: 0x9873, mtu: 130687]
SingleFrame **Version:** Highest version supported by both Core and Proxy **Payload:** Lots of bytes	————>	

4.2.2 STARTING COMMUNICATION

The application sends a StartService frame to the module containing no payload.

APPLICATION -> HEAD UNIT

4.2.2.1 VERSIONS 1 - 4

Version	C	Frame Type	Service Type	Frame Info	Session Id	Data Size
1	no	Control	RPC	Start Service	0	0
0b0001	0b0	0b000	0x07	0x01	0x00	0x00000000

4.2.2.2 VERSIONS 5 AND NEWER

Note: Even though this is a Protocol Version 1 frame header it includes a payload. This is a very special exception.

Payload includes a constructed BSON object that has a single parameter of `protocolVersion` that describes the applications max supported Protocol Version.

Version	C	Frame Type	Service Type	Frame Info	Session Id	Data Size
1	no	Control	RPC	Start Service	0	Size of Payload
0b0001	0b0	0b000	0x07	0x01	0x00	0xNNNNNNNN

Payload
[protocolVersion: x.x.x]

4.2.3 SUCCESS

If the head unit allows the RPC service to start, it will respond with a StartServiceACK . At this time the version will finish its negotiation process.

HEAD UNIT -> APPLICATION
4.2.3.1 PROTOCOL VERSIONS 1-4

The StartServiceACK will contain the module's maximum supported protocol version. The packet structure will also match that of the supplied version; if the module's maximum supported version is 1, the packet will contain an 8 byte header (version 1), otherwise it will contain a 12 byte header (version 2). The application will then find the highest version supported by both the module and the application. This will be the determined version used for this session and will be used for all other packets sent from this point forward as well as all other services.

The payload of the StartServiceACK will contain a hash of the service which was started on the head unit if the payload size is greater than 0. This hash should be stored by an application and is needed in the end communication flow.

Version	E	Frame Type	Service Type	Frame Info	Session Id	Data Size	Message ID
Max Module Version (4)	no	Control	RPC	Start Service ACK	Assigned Session	4	2
0b0100	0b0	0b000	0x07	0x02	0x01	0x00000004	0x000000000n

4.2.3.2 PROTOCOL VERSIONS 5 AND NEWER

The `StartServiceACK` will contain a negotiated version between what the application provided in the `StartService` frame and what the module's maximum supported protocol version. Thus, if it is determined that no such information was sent in the `StartService` frame, the module will assume the previous method of version negotiation and send version 4 to assume it's max version supplied to the application and wait for the incoming `RegisterAppInterface` RPC from the application to finally determine the negotiated version. Either way, the determined version will be used for this session including all other packets sent from this point forward as well as all other services.

The packet structure will also match that of the supplied version; if the module's maximum supported version is 1, the packet will contain an 8 byte header (version 1), otherwise it will contain a 12 byte header (version 2). If the protocol version was determined to be 5 or higher, the payload it contains will be constructed in nature as a BSON object. The application will then find the highest version supported by both the module and the application. This will be the determined version used for this session and will be used for all other packets sent from this point forward as well as all other services.

The payload of the `StartServiceACK` will contain the agreed upon full protocol version *"Major.Minor.Patch"*, a hash of the service which was started on the head unit, and the max transport unit for that session (0x07 RPC). The hash should be stored by an application and is needed in the end communication flow. The MTU should be used as the default

MTU for all other services for that session unless otherwise provided in the corresponding StartServiceACK for that service.

4.2.3.2.1 Protocol Version Supplied in StartService

Version	E	Frame Type	Service Type	Frame Info	Session Id	Data Size	Message ID
Max major version supported by module and application	no	Control	RPC	Start Service ACK	Assigned Session	Size of payload	2
0bNNNN	0b0	0b000	0x07	0x02	0x01	0xNNNNNNNN	0x0000000n

Payload
[protocolVersion: x.x.x, hashId: 0xNNNN, mtu: 130687]

4.2.3.2.2 Protocol Version Not Supplied in StartService

Version	E	Frame Type	Service Type	Frame Info	Session Id	Data Size	Message ID
Max version application can possibly support (4)	no	Control	RPC	Start Service ACK	Assigned Session	4	2
0b0100	0b0	0b000	0x07	0x02	0x01	0x00000004	0x000000000n

4.2.4 FAILURE

If a session has already been started, or can't be started, a `StartServiceNAK` will be sent in response to the `StartService` packet.

HEAD UNIT -> APPLICATION

4.2.4.1 PROTOCOL VERSIONS 1 THROUGH 4

Version	E	Frame Type	Service Type	Frame Info	Session Id	Data Size	Message ID
Max Module Version (4)	no	Control	RPC	Start Service NAK	0	0	0
0b0100	0b0	0b000	0x07	0x03	0x00	0x00000000	0x0000000000

4.2.4.1 PROTOCOL VERSIONS 5 AND NEWER

Version	E	Frame Type	Service Type	Frame Info	Session Id	Data Size	Message ID
Max Module Version (4)	no	Control	RPC	Start Service NAK	0	Size of Payload	0
0b0100	0b0	0b000	0x07	0x03	0x00	0xNNNNNNNN	0x0000000000

Payload
[rejectedParams:[protocolVersion, x, x]]

The RPC service always needs to be started as unencrypted first, then it can be moved to an encrypted state by sending another `StartService` request containing an encryption flag set to `1` at a later point. Services of another type can be started as encrypted initially, i.e. it is not necessary to start them as unencrypted and then move to encrypted state using second `StartService` request (however such sequence of actions is also valid). See "7. Secured Communication" section for more details.

4.3 Registration

> Required: All Protocol Versions

Each application registers for continued communication with the head unit by sending a `RegisterAppInterface` Request RPC to the head unit via the RPC Service. Additional services can only be started after a successful `RegisterAppInterface` Response RPC has been sent from the head unit to the application.

4.4 Starting other services

While the RPC service is the default service that is started to establish a connection and a session, the application may wish to start other services. Similar to the process in Section 4, all services that are to be to started in a session require a `StartService` packet to be sent from the application. If the module supports and allows that service type to be started, it will respond with a `StartServiceACK` that has a payload of the hash ID for that service. If the module is unable to start that service or that application does not have access to that service, it will respond with a `StartServiceNAK`.

4.5 Heartbeat

> **Deprecated: Protocol Versions 4 and higher**

> Required: Protocol Version 3

> Added: Protocol Version 3

After a successful start service exchange between the application and head unit both the application and head unit are required to be able to respond to heartbeat messages if the negotiated protocol version is 3. After sending a heartbeat, if the application or head unit

does not respond within a timeout (custom per app/head unit), the sender will disconnect. The sender's timer for the heartbeat timeout should be reset every time any message is received. Heartbeats are sent using the Control Service Type (0x00)

4.5.1 HEARTBEAT REQUEST

Note: The request can originate from either the Head Unit or the Application

HEAD UNIT -> APPLICATION

Version	E	Frame Type	Service Type	Frame Info	Session Id	Data Size	Message ID
4	no	Control	Control	Heartbeat	0	0	0
0b0100	0b0	0b000	0x00	0x00	0x00	0x00000000	0x0000000000

4.5.2 HEARTBEAT ACK

Note: The response ACK will originate from the Head Unit or the Application based on the origin of the request

APPLICATION -> HEAD UNIT

Vers ion	E	Frame Type	Servic e Type	Frame Info	Sessio n Id	Data Size	Mes sage ID
4	no	Control	Control	Heartbe at ACK	0	0	0
0b01 00	0b0	0b000	0x00	0xFF	0x00	0x0000 0000	0x00 0000 00

4.5.3 HEARTBEAT NAK

There is currently no heartbeat NAK.

4.6 Secondary Transport

Added: Protocol version 5.1.0

After the RPC service has been established on an initial transport, it is possible to utilize a different transport beyond the initial transport for certain services. This additional transport is called the "Secondary Transport". The initial transport used to start the RPC service is called the "Primary Transport".

4.6.1 SECONDARY TRANSPORT REGISTRATION

The RPC `StartServiceACK` will include information on potential Secondary Transports in the parameter `secondaryTransports` if any are supported. Once received, it is possible to register the session on the Secondary Transport if connected; if the transport is not connected it will have to either wait until an update is received through the `TransportEven tUpdated` frame or the physical connection is made.

Once the connection for Secondary Transport is established, if the application wishes to utilize that transport as a SecondaryTransport, the application is required to send a `RegistererSecondaryTransport` frame on that transport. The head unit will respond with either a `RegisterSecondaryTransportACK` or `RegisterSecondaryTransportNAK` frame. If the registration was successful and the application receives a `RegisterSecondaryTransportACK`, it may then utilize the Secondary Transport to start services.

4.6.1.1 REGISTERSECONDARYTRANSPORT

APPLICATION -> HEAD UNIT

Version	E	Frame Type	Service Type	Frame Info	Session Id	Data Size	Message ID
Max major version supported by module and application	no	Control	Control	Register Secondary Transport	Session Id assigned on Primary Transport	0	1
0bNNNN	0b0	0b000	0x00	0x07	0xNN	0x0000 0000	0x00 0000 0001

4.6.1.2 REGISTERSECONDARYTRANSPORTACK

HEAD UNIT -> APPLICATION

Version	E	Frame Type	Service Type	Frame Info	Session Id	Data Size	Message ID
Max major version supported by module and application	no	Control	Control	Register Secondary Transport ACK	Session Id assigned on Primary Transport	0	2
0bNNNN	0b0	0b000	0x00	0x08	0xNN	0x00000000	0x0000000002

4.6.1.3 REGISTERSECONDARYTRANSPORTNAK

HEAD UNIT -> APPLICATION

Version	E	Frame Type	Service Type	Frame Info	Session Id	Data Size	Message ID
Max major version supported by module and application if known, otherwise 5	no	Control	Control	Register Secondary Transport NAK	Session Id assigned on Primary Transport	0	2
0bNNNN	0b0	0b000	0x00	0x09	0xNN	0x00000000	0x0000000002

4.6.2 TRANSPORT EVENT UPDATE

Some Secondary Transports might require additional details on how they can be established. For example, in order to establish a TCP connection between the application and head unit the IP address and port number are required. The `TransportEventUpdate` control frame is used for this purpose.

The head unit will send out a `TransportEventUpdate` frame whenever its transport configuration is changed, for example when its IP address is updated or Wi-Fi network

goes down. The head unit will also send out a TransportEventUpdate frame right after the StartServiceACK frame that establishes the RPC service so the application can initiate a connection immediately.

The TransportEventUpdate frame must always be sent through the Primary Transport. The head unit should not send the frame to applications that don't support protocol versions 5.1.0 or newer.

4.6.2.1 TRANSPORTEVENTUPDATE
HEAD UNIT -> APPLICATION

Version	E	Frame Type	Service Type	Frame Info	Session Id	Data Size	Message ID
Max major version supported by module and application	no	Control	Control	Transport Event Update	Session Id assigned on Primary Transport	Size of payload	Variable
0bNNNN	0b0	0b000	0x00	0xFD	0xNN	0xNNNNNNNN	0xNNNNNNNN

Payload
[tcpIpAddress:"x.x.x.x", tcpPort:NNNN]

4.6.3 STARTING SERVICES ON SECONDARY TRANSPORTS

A Secondary Transport is capable of carrying the video and audio services. Other services, including RPC and Hybrid service, must always run on the Primary Transport. (Note: Control service is an inherently started service on a transport and does not need to be established, but frames will be handled on a frame by frame basis of Primary vs Secondary Transport support)

The RPC `StartServiceACK` might include the parameters `audioServiceTransports` and `videoServiceTransports` describing which service is allowed to run on which transport priority type (Primary, Secondary or both). An application honors this information and starts the service(s) only on an allowed transport. For example, if video service is allowed only on a Secondary Transport, the application will not start video streaming until Secondary Transport is established and registered.

The transport priority types included in these parameters are listed in preferred order, for example, `[2,1]` (Secondary , Primary). In this case the priority of the Secondary Transport is higher than that of Primary Transport, the application may stop and restart service(s) when the Secondary Transport is added or removed. However, each service type must only be started and carried on a single transport at a time.

When starting a service over a Secondary Transport the application, it must follow the previous sections to establish the transport connection and register its session over that transport. At that point it runs the normal sequence described in section 4.4. When starting a service over a Secondary Transport, the session ID that was provided during the establishment of the RPC service should be used.

4.6.4 TERMINATING SECONDARY TRANSPORT

There is no procedure to terminate a Secondary Transport. However, if the Primary Transport is disconnected or the RPC service is stopped, any Secondary Transport for that session should be unregistered and if no other sessions are registered over that Secondary Transport it should be disconnected.

5. Services

Every active session has the ability to start any of the services defined in this protocol spec as long as they have permission on the module in which they are connected. Every session can only have one of each type of service open at a time.

Messages sent have a priority based on their Service Type. Lower values for service type have higher delivery priority. A message's payload's format is based on the different service types defined below.

5.1 Control Service

> Required: All Protocol Versions

The control service is the lowest level service available. While Control Frame packets are used frequently, the control service itself is rarely used.

5.1.1 SECURITY QUERY

When establishing a secure connection, the TLS Payload is sent in a control service message with a binary query header. The size of this header is 12 bytes, similar to the RPC Payload Binary Header.

5.1.1.1 PAYLOAD

The security query is able to contain JSON data as well as binary data. During the handshake the TLS handshake data is sent as binary data. See "Send Handshake Data" section for details.

In case of an error, a notification is sent with an error code and error message as JSON data. See "Send Internal Error" section for details.

Binary Query Header
JSON Data
Binary Data

5.1.1.2 BINARY HEADER

Byte 1	Byte 2	Byte 3	Byte 4
Query Type		Query ID	
Sequential Number			
JSON Size			

5.1.1.2.1 Binary Header Fields

Field	Size	Description
Query Type	8 bit	0x00 Request 0x01 - 0x0F Reserved 0x10 Response 0x11 - 0x1F Reserved 0x20 Notification 0x21 - 0xFE Reserved 0xFF Invalid Query Type
Query ID	24 bit	0x000001 Send Handshake Data 0x000002 Send Internal Error 0x000003 - 0xFFFFFE Reserved 0xFFFFFF Invalid Query ID
Sequential Number	32 bit	Message ID can be set by the mobile libraries to track security messages. The system uses the same Message ID when replying to the query allowing the mobile libraries to correlate messages.
JSON Size	32 bit	The size of the JSON data following the binary query header. Any additional data following the JSON data in the payload is binary data.

5.1.1.3 SEND HANDSHAKE DATA

When performing the TLS handshake, the server sends a "Send Handshake Data" request containing its handshake data to the client, and the client sends a response with its own handshake data accordingly.

5.1.1.3.1 Request Payload

Note: Prior to SDL Core version 8.0.0, Core sent the "Notification" type (0x20) in this message instead of "Request" (0x00). Client libraries should account for this when communicating with older versions of SDL Core.

Binary Query Header

Query Type	TLS Message Type	Sequential Number	JSON Size
Request	Send Handshake Data	Any number to be used to correlate query messages	Zero
0x00	0x000001	0xNNNNNNNN	0x00000000

Binary Data: SSL Handshake Request

5.1.1.3.2 Response Payload

Binary Query Header			

Query Type	TLS Message Type	Sequential Number	JSON Size
Response	Send Handshake Data	Any number to be used to correlate query messages	Zero
0x10	0x000001	0xNNNNNNNN	0x00000000
Binary Data: SSL Handshake Response			

5.1.1.4 SEND INTERNAL ERROR

If an error occurs during the TLS handshake, a notification is sent with both JSON data and binary data describing the error. The JSON data contains the error code and an error text. The binary data is one single byte and only contains the error code.

The error code in JSON data and the binary data are the same value from the same code list.

5.1.1.4.1 PAYLOAD

The following query header is used by the system and the application to send error messages.

Binary Query Header			

Query Type	TLS Message Type	Sequential Number	JSON Size
Notification	Send Internal Error	Unused	Size of the JSON data
0x20	0x000002	0xNNNNNNNN	0xNNNNNNNN
JSON Data			
Binary Data: Single Byte Error Code			

5.1.1.4.2 JSON STRUCTURE

Key	Description
id	A decimal value representing an Error code.
text	A string describing the error.

5.1.1.4.3 ERROR CODES

Error code	Byte	Value	Description
ERROR_SUCCESS	0x00	0	Internal Security Manager value
ERROR_INVALID_QUERY_SIZE	0x01	1	Wrong size of query data
ERROR_INVALID_QUERY_ID	0x02	2	Unknown Query ID
ERROR_NOT_SUPPORTED	0x03	3	SDL does not support encryption
ERROR_SERVICE_ALREADY_PROTECTED	0x04	4	Received request to protect a service that was protected before
ERROR_SERVICE_NOT_PROTECTED	0x05	5	Received handshake or encrypted data for not protected service
ERROR_DECRYPTION_FAILED	0x06	6	Decryption failed
ERROR_ENCRYPTION_FAILED	0x07	7	Encryption failed
ERROR_SSL_INVALID_DATA	0x08	8	SSL invalid data
ERROR_HANDSHAKE_FAILED	0x09	9	In case of all other handshake errors

INVALID_CERT	0x0A	10	Handshake failed because certificate is invalid
EXPIRED_CERT	0x0B	11	Handshake failed because certificate is expired
ERROR_INTERNAL	0xFF	255	Internal error
ERROR_UNKNOWN _INTERNAL_ERROR	0xFE	254	Error value for testing

5.2 RPC Service

Required: All Protocol Versions

The RPC service is used to send requests, responses, and notifications between an application and a head unit. Valid messages are defined in the RPC Specification.

The payload of a message sent via the RPC service, which directly follows the Frame Header in the packet, consists of a Binary Header, and JSON data representing the RPC.

RPC PAYLOAD

5.2.1 BINARY HEADER

Required: Protocol Version 2 and greater

Byte 1	Byte 2	Byte 3	Byte 4
RPC Type	RPC Function ID		
Correlation ID			
JSON Size			

5.2.1.1 BINARY HEADER FIELDS

Field	Size	Description
RPC Type	4 bit	0x0 Request 0x1 Response 0x2 Notification 0x3 Erroneous Response 0x4 - 0xF Reserved
RPC Function ID	28 bit	The Function ID of each RPC is specific to each version of the RPC Specification but in general do not change from version to version.
Correlation ID	32 bits (signed)	The Correlation ID is used to map a request to its response. Requests sent in the same session with the same Correlation ID as a pending request will be rejected with an `INVALID_ID` response. Requests that use a Correlation ID less than 0 will be rejected with an `INVALID_ID` response. In Protocol Version 1, when the Binary Header did not exist, the Correlation ID was included as part of the JSON and has a max value of 65536.
JSON Size	32 bits	The size of the JSON Data following the Binary Header in the RPC Payload

5.3 Hybrid (Bulk Data) Service

> Required: Protocol Version 2 and greater

The Hybrid Service does not need to be explicitly started; all applications that have successfully started the RPC Service have access to the Hybrid Service.

The Hybrid Service is similar to the RPC Service but adds a bulk data field. The payload of a message sent via the Hybrid service consists of a Binary Header, JSON Data, and Bulk Data.

The size of the Bulk Data field is the Data Size (Found in the Frame Header) minus the 12 Bytes of the Binary Header minus the JSON Size (Found in the Binary Header).

The Binary Header of a message using the Hybrid Service is the same as the Binary Header of a message using the RPC Service.

HYBRID SERVICE PAYLOAD

5.4 Audio Service (PCM)

> Available: Protocol Version 3 and greater

The application can start the audio service to send PCM audio data to the head unit. After the `StartService` packet is sent and the ACK received, the payload for the Audio Service is only PCM audio data.

5.5 Video Service (H.264)

> Available: Protocol Version 3 and greater

The application can start the video service to send H.264 video data to the head unit. After the `StartService` packet is sent and the ACK received, the payload for the Video Service is only H.264 video data.

6. Ending Communication

The application may request it's session to be ended outside of a transport disconnect, module power cycle, etc.

6.1 Completely Closing a Session and Ending All Services

To close out a communication session with the head unit, an application sends an `EndService` packet with service type 7 (RPC) to the module. The `EndService` packet payload should include the correct hash ID supplied with the `StartServiceACK`.

6.2 Closing Specific Services

If the application doesn't want to completely stop its session, but only wishes to close a specific session it can do so using an `EndService` packet that's service type matches the service that the application is trying to close. The `EndService` packet should include the hash ID in its payload that was contained in the `StartServiceACK` for that specific service.

7. Secured Communication

It is possible to establish a secured and encrypted communication with the system by setting the frame header encryption flag to `1` when starting a new service or by sending another `StartService` with the encryption flag set to `1` when the service is already established (this the required flow for the RPC service). If the authentication is successful, the system will reply with a `StartService ACK` frame with the encryption flag also set to `1` indicating that encrypted data is now accepted. If the authentication fails for some reason, the system will reset the TLS connection and return a `StartService NAK` frame.

Below are possible combinations of the service encryption status and RPCs protection flag value.

Service Encryption Status	RPC Type	Requires Protection	Expected SDL Behavior
Encryption is not established	Request	yes	SDL Core rejects the request with result code `ENCRYPTION_NEEDED` (please see policy updates for which RPCs need protection).
		no	SDL Core continues processing the RPC request.
	Notification	yes	SDL Core does not send the notification.
		no	SDL Core sends the notification unencrypted.
Encryption is established	Request	yes	If unencrypted, SDL Core rejects the request with an unencrypted response and result code `ENCRYPTION_NEEDED`. If encrypted, SDL Core continues processing the request and sends an encrypted response.
		no	If unencrypted, SDL Core continues processing the request and sends an unencrypted response. If encrypted, SDL Core continues processing the request and sends an encrypted response.
	Notification	yes	SDL Core sends the notification encrypted.
		no	SDL Core sends the notification unencrypted.

7.1 Authentication

The authentication is done using TLS handshake. The TLS handshake process is defined by TLS and is not part of the SDL protocol.

The below diagram shows the sequence of how the TLS handshake exchanges certificates to compute the master secret.
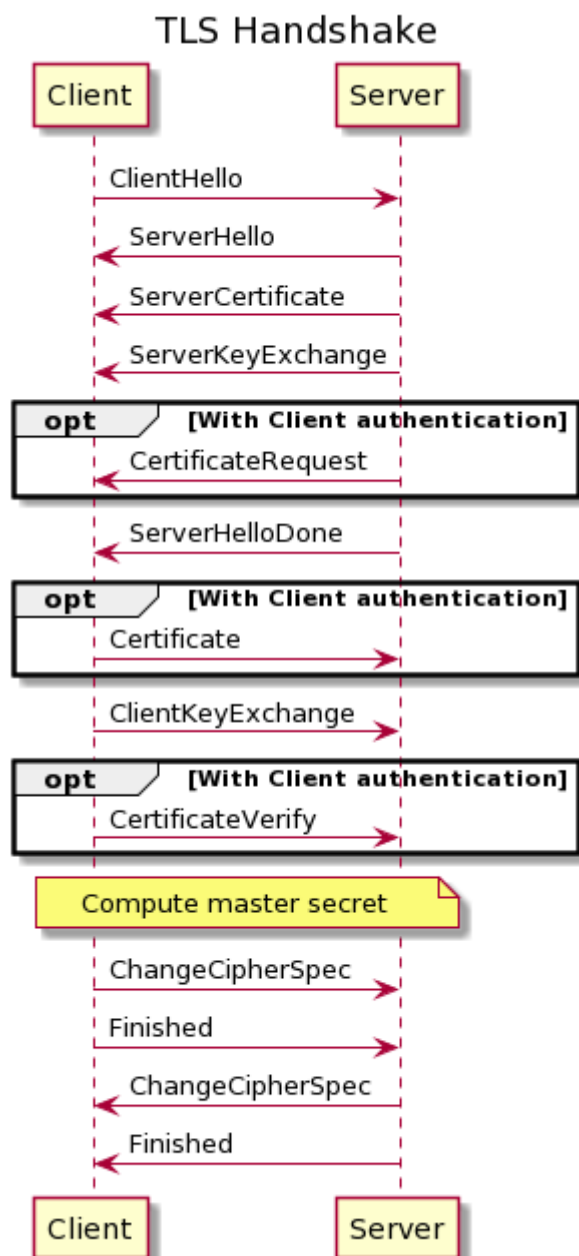
Please see SDL Overview Guides for more details.

The system can be configured to support one encryption method. The following methods are supported:

- TLSv1
- TLSv1.1
- TLSv1.2
- DTLSv1
- SSLv3 (not supported on most newer systems)

The system has to initiate with the corresponding client method. For instance, if the system is configured to use `DTLSv1`, it has to use the method `DTLSv1_client`. The application role has to be server and must use `DTLSv1_server`.

The system also supports configurable SSL Security level introduced in OpenSSL 1.1.0. This parameter can be changed by `SecurityLevel` parameter in the Core configuration file. By default, system uses security level 1 for TLS handshakes. At this time setting the security level higher than 1 for general internet use is likely to cause considerable interoperability issues and is not recommended. This is because the SHA1 algorithm is very widely used in certificates and will be rejected at levels higher than 1 because it only offers 80 bits of security.

7.2 Handshake Frames

The system will initiate a TLS handshake to authenticate the application where the system's role will be the client while the application's role will be the server. The system will do this only once if the application was not authenticated before in the current transport connection. The TLS handshake data is always sent in single frames. The service type for TLS handshake is the control service.

7.2.1 SDL PROTOCOL FRAME HEADER

The following SDL frame header is used for every frame related to TLS handshake.

SDL Protocol Frame Header

Version	E	Frame Type	Service Type	Frame Info	Session ID	Data Size	Message ID
Max major version supported by module and application	no	Single Frame	Control Service	Single Frame Info	Assigned Session ID	Query Binary Header + JSON Data size + Binary Handshake Data size	Enumerated number
0xN	0b0	0b001	0x00	0x00	0xNN	0xC + 0xNNNNNNNN + 0xNNNNNNNN	0xNNNNNNNN

7.2.2 SECURITY QUERY BINARY HEADER

The following query header is used by the system and the application to send TLS handshake data.

Binary Query Header			

Query Type	TLS Message Type	Sequential Number	JSON Size
Request	Send Handshake Data	Any number to be used to correlate query messages	Zero
0x00	0x000001	0xNNNNNNNN	0x00000000
Binary TLS Handshake Data			

7.3 Error handling

In case of an error, the system and the application should reset the active SSL connection of the current transport connection. This impacts already established secured service sessions as all of them will be closed. The application will need to restart all services which require protection.

SmartDeviceLink

RPC Spec

Version: 8.0.0

Enumerations

Result

ELEMENTS

VALUE	DESCRIPTION
SUCCESS	The request succeeded
UNSUPPORTED_REQUEST	The request is not supported by the headunit
UNSUPPORTED_RESOURCE	One or more of the items (phoneme type, button name, image type, etc.) in the request is not supported by the HMI.
DISALLOWED	RPC is not authorized in local policy table.
REJECTED	The requested command was rejected, e.g. because mobile app is in background and cannot perform any HMI commands. Or an HMI command (e.g. Speak) is rejected because a higher priority HMI command (e.g. Alert) is playing.
ABORTED	A command was aborted, for example due to user interaction (e.g. user pressed button). Or an HMI command (e.g. Speak) is aborted because a higher priority HMI command (e.g. Alert) was requested.
IGNORED	A command was ignored, because the intended result is already in effect. For example, SetMediaClockTimer was used to pause the media clock although the clock is paused already. NOTE: potentially replaces SUBSCRIBED_ALREADY
RETRY	The user interrupted the RPC (e.g. PerformAudioPassThru) and indicated to start over. Note, the app must issue the new RPC.

VALUE	DESCRIPTION
IN_USE	The data may not be changed, because it is currently in use. For example when trying to delete a command set that is currently involved in an interaction.
VEHICLE_DATA_NOT_AVAILABLE	The requested vehicle data is not available on this vehicle or is not published.
TIMED_OUT	Overlay reached the maximum timeout and closed.
INVALID_DATA	The data sent is invalid. For example: Invalid Json syntax Parameters out of bounds (number or enum range) Mandatory parameters not provided Parameter provided with wrong type Invalid characters Empty string
CHAR_LIMIT_EXCEEDED	
INVALID_ID	One of the provided IDs is not valid. For example This applies to CorrelationID, SubscriptionID, CommandID, MenuID, etc.
DUPLICATE_NAME	There was a conflict with an registered name (application or menu item) or vr command
APPLICATION_NOT_REGISTERED	An command can not be executed because no application has been registered with RegisterApplication.
WRONG_LANGUAGE	The requested language is currently not supported. Might be because of a mismatch of the currently active language on the headunit and the requested language

VALUE	DESCRIPTION
OUT_OF_MEMORY	The system could not process the request because the necessary memory couldn't be allocated
TOO_MANY_PENDING_REQUESTS	There are too many requests pending (means, that the response has not been delivered, yet).There may be a maximum of 1000 pending requests at a time.
TOO_MANY_APPLICATIONS	There are already too many registered applications
APPLICATION_REGISTERED_ALREADY	RegisterApplication has been called again, after a RegisterApplication was successful before.
WARNINGS	The RPC (e.g. SubscribeVehicleData) executed successfully but one or more items have a warning or failure.
GENERIC_ERROR	Provided data is valid but something went wrong in the lower layers.
USER_DISALLOWED	RPC is included in a functional group explicitly blocked by the user.
TRUNCATED_DATA	The RPC (e.g. ReadDID) executed successfully but the data exceeded the platform maximum threshold and thus, only part of the data is available.
UNSUPPORTED_VERSION	Sync doesn't support the protocol that is requested by the mobile application

VALUE	DESCRIPTION
VEHICLE_DATA_NOT_ALLOWED	The user has turned off access to vehicle data, and it is globally unavailable to mobile applications.
FILE_NOT_FOUND	A specified file could not be found on the headunit.
CANCEL_ROUTE	User selected to Cancel Route.
SAVED	The RPC (e.g. Slider) executed successfully and the user elected to save the current position / value.
INVALID_CERT	The certificate provided during authentication is invalid.
EXPIRED_CERT	The certificate provided during authentication is expired.
RESUME_FAILED	The provided hash ID does not match the hash of the current set of registered data or the core could not resume the previous data.
DATA_NOT_AVAILABLE	The requested information is currently not available. This is different than UNSUPPORTED_RESOURCE because it implies the data is at some point available.
READ_ONLY	The value being set is read only
CORRUPTED_DATA	The data sent failed to pass CRC check in receiver end
VALUE	DESCRIPTION

VALUE	DESCRIPTION
ENCRYPTION_NEEDED	SDL receives an un-encrypted RPC request that needs protection.

ButtonPressMode

ELEMENTS

VALUE	DESCRIPTION
LONG	A button was released, after it was pressed for a long time Actual timing is defined by the headunit and may vary
SHORT	A button was released, after it was pressed for a short time Actual timing is defined by the headunit and may vary

ButtonEventMode

ELEMENTS

VALUE	DESCRIPTION
BUTTONUP	A button has been released up
BUTTONDOWN	A button has been pressed down

Language

ELEMENTS

VALUE	DESCRIPTION
EN-US	English - US
ES-MX	Spanish - Mexico
FR-CA	French - Canada
DE-DE	German - Germany
ES-ES	Spanish - Spain
EN-GB	English - GB
RU-RU	Russian - Russia
TR-TR	Turkish - Turkey
PL-PL	Polish - Poland
FR-FR	French - France
IT-IT	Italian - Italy
SV-SE	Swedish - Sweden
PT-PT	Portuguese - Portugal
NL-NL	Dutch (Standard) - Netherlands
EN-AU	English - Australia
VALUE	DESCRIPTION

VALUE	DESCRIPTION
ZH-CN	Mandarin - China
ZH-TW	Mandarin - Taiwan
JA-JP	Japanese - Japan
AR-SA	Arabic - Saudi Arabia
KO-KR	Korean - South Korea
PT-BR	Portuguese - Brazil
CS-CZ	Czech - Czech Republic
DA-DK	Danish - Denmark
NO-NO	Norwegian - Norway
NL-BE	Dutch (Flemish) - Belgium
EL-GR	Greek - Greece
HU-HU	Hungarian - Hungary
FI-FI	Finnish - Finland
SK-SK	Slovak - Slovakia
EN-IN	English - India

VALUE	DESCRIPTION
TH-TH	Thai - Thailand
EN-SA	English - Middle East
HE-IL	Hebrew - Israel
RO-RO	Romanian - Romania
UK-UA	Ukrainian - Ukraine
ID-ID	Indonesian - Indonesia
VI-VN	Vietnamese - Vietnam
MS-MY	Malay - Malaysia
HI-IN	Hindi - India

UpdateMode

Describes how the media clock timer should behave on the platform

ELEMENTS

VALUE	DESCRIPTION
COUNTUP	Starts the media clock timer counting upwards, as in time elapsed.
COUNTDOWN	Starts the media clock timer counting downwards, as in time remaining.
PAUSE	Pauses the media clock timer
RESUME	Resume the media clock timer
CLEAR	Clears the media clock timer (previously done through Show->mediaClock)

TimerMode

ELEMENTS

VALUE	DESCRIPTION
UP	Causes the media clock timer to update from 0:00 to a specified time
DOWN	Causes the media clock timer to update from a specified time to 0:00
NONE	Indicates to not use the media clock timer

InteractionMode

For application-requested interactions, this mode indicates the method in which the user is notified and uses the interaction.

VALUE	DESCRIPTION
MANUAL_ONLY	This mode causes the interaction to only occur on the display, meaning the choices are provided only via the display. No Voice Interaction.
VR_ONLY	This mode causes the interaction to only occur using the headunits VR system. Selections are made by saying the command.
BOTH	This mode causes both a VR and display selection option for an interaction. The user will first be asked via Voice Interaction (if available). If this is unsuccessful, the system will switch to manual input.

LayoutMode

For touchscreen interactions, the mode of how the choices are presented.

ELEMENTS

VALUE	DESCRIPTION
ICON_ONLY	This mode causes the interaction to display the previous set of choices as icons.
ICON_WITH_SEARCH	This mode causes the interaction to display the previous set of choices as icons along with a search field in the HMI.
LIST_ONLY	This mode causes the interaction to display the previous set of choices as a list.
LIST_WITH_SEARCH	This mode causes the interaction to display the previous set of choices as a list along with a search field in the HMI.
KEYBOARD	This mode causes the interaction to immediately display a keyboard entry through the HMI.

WindowType

ELEMENTS

VALUE	DESCRIPTION
MAIN	This window type describes the main window on a display.
WIDGET	A widget is a small window that the app can create to provide information and soft buttons for quick app control.

PredefinedWindows

Specifies IDs for windows which are predefined and pre-created. The mobile libraries and SDL Core use the integer value when referencing these elements.

ELEMENTS

VALUE	DESCRIPTION
DEFAULT_WINDOW	The default window is a main window pre-created on behalf of the app.
PRIMARY_WIDGET	The primary widget of the app.

HMILevel

Enumeration that describes current levels of HMI.

ELEMENTS

VALUE	DESCRIPTION
FULL	
LIMITED	
BACKGROUND	
NONE	

AudioStreamingState

Enumeration that describes possible states of audio streaming.

ELEMENTS

VALUE	DESCRIPTION
AUDIBLE	
ATTENUATED	
NOT_AUDIBLE	

SystemAction

Enumeration that describes system actions that can be triggered.

ELEMENTS

VALUE	DESCRIPTION
DEFAULT_ACTION	Default action occurs. Standard behavior (e.g. SoftButton clears overlay).
STEAL_FOCUS	App is brought into HMI_FULL.
KEEP_CONTEXT	Current system context is maintained. An overlay is persisted even though a SoftButton has been pressed and the notification sent.

SystemContext

Enumeration that describes possible contexts an app's HMI might be in. Communicated to whichever app is in HMI FULL, except Alert.

ELEMENTS

VALUE	DESCRIPTION
MAIN	The app's persistent display (whether media/non-media/navigation) is fully visible onscreen.
VRSESSION	The system is currently in a VR session (with whatever dedicated VR screen being overlaid onscreen).
MENU	The system is currently displaying an in-App menu onscreen.
HMI_OBSCURED	The app's display HMI is currently being obscured by either a system or other app's overlay.
ALERT	Broadcast only to whichever app has an alert currently being displayed.

VideoStreamingState

Enumeration that describes possible states of video streaming.

ELEMENTS

VALUE	DESCRIPTION
STREAMABLE	
NOT_STREAMABLE	

SoftButtonType

Contains information about the SoftButton capabilities.

ELEMENTS

VALUE	DESCRIPTION
TEXT	
IMAGE	
BOTH	

AppInterfaceUnregisteredReason

Error code, which comes from the module side.

ELEMENTS

VALUE	DESCRIPTION
IGNITION_OFF	
BLUETOOTH_OFF	
USB_DISCONNECTED	
REQUEST_WHILE_IN_NONE_HMI_LEVEL	
TOO_MANY_REQUESTS	
DRIVER_DISTRACTION_VIOLATION	
LANGUAGE_CHANGE	
MASTER_RESET	
FACTORY_DEFAULTS	
APP_UNAUTHORIZED	
PROTOCOL_VIOLATION	
UNSUPPORTED_HMI_RESOURCE	
RESOURCE_CONSTRAINT	By sending this value, SDL unregisters the application to allow the HMI to close the application.

TriggerSource

Indicates the source from where the command was triggered.

ELEMENTS

VALUE	DESCRIPTION
MENU	
VR	
KEYBOARD	

HmiZoneCapabilities

Contains information about the HMI zone capabilities. For future use.

ELEMENTS

VALUE	DESCRIPTION
FRONT	
BACK	

SpeechCapabilities

Contains information about the TTS capabilities.

ELEMENTS

VALUE	DESCRIPTION
TEXT	
SAPI_PHONEMES	
LHPLUS_PHONEMES	
PRE_RECORDED	
SILENCE	
FILE	

VrCapabilities

Contains information about the VR capabilities.

ELEMENTS

VALUE	DESCRIPTION
TEXT	

PrerecordedSpeech

Contains a list of prerecorded speech items present on the platform.

ELEMENTS

VALUE	DESCRIPTION
HELP_JINGLE	
INITIAL_JINGLE	
LISTEN_JINGLE	
POSITIVE_JINGLE	
NEGATIVE_JINGLE	

SamplingRate

Describes different sampling options for PerformAudioPassThru.

ELEMENTS

VALUE	DESCRIPTION
8KHZ	Sampling rate of 8000 Hz.
16KHZ	Sampling rate of 16000 Hz.
22KHZ	Sampling rate of 22050 Hz.
44KHZ	Sampling rate of 44100 Hz.

BitsPerSample

Describes different quality options for PerformAudioPassThru.

ELEMENTS

VALUE	DESCRIPTION
8_BIT	Audio sample is 8 bits wide, unsigned.
16_BIT	Audio sample is 16 bits wide, signed, and in little endian.

AudioType

Describes different audio type options for PerformAudioPassThru.

ELEMENTS

VALUE	DESCRIPTION
PCM	Linear PCM.

VehicleDataType

Defines the data types that can be published and subscribed to.

ELEMENTS

VALUE	DESCRIPTION
VEHICLEDATA_GPS	Notifies GPSData may be subscribed
VEHICLEDATA_SPEED	
VEHICLEDATA_RPM	
VEHICLEDATA_FUELLEVEL	
VEHICLEDATA_FUELLEVEL_STATE	
VEHICLEDATA_FUELCONSUMPTION	
VEHICLEDATA_CLIMATEDATA	
VEHICLEDATA_EXTERNTEMP	
VEHICLEDATA_VIN	
VEHICLEDATA_GEARSTATUS	
VEHICLEDATA_PRNDL	
VEHICLEDATA_TIREPRESSURE	
VEHICLEDATA_ODOMETER	
VEHICLEDATA_BELTSTATUS	
VEHICLEDATA_BODYINFO	

VALUE	DESCRIPTION

VALUE	DESCRIPTION
VEHICLEDATA_DEVICESTATUS	
VEHICLEDATA_ECALLINFO	
VEHICLEDATA_AIRBAGSTATUS	
VEHICLEDATA_EMERGENCYEVENT	
VEHICLEDATA_CLUSTERMODESTATUS	
VEHICLEDATA_MYKEY	
VEHICLEDATA_BRAKING	
VEHICLEDATA_WIPERSTATUS	
VEHICLEDATA_HEADLAMPSTATUS	
VEHICLEDATA_BATTVOLTAGE	
VEHICLEDATA_ENGINETORQUE	
VEHICLEDATA_ACCPEDAL	
VEHICLEDATA_STEERINGWHEEL	
VEHICLEDATA_TURNSIGNAL	
VEHICLEDATA_FUELRANGE	

VALUE	DESCRIPTION
VEHICLEDATA_ENGINEOILLIFE	
VEHICLEDATA_ELECTRONICPARKBRAKESTATUS	
VEHICLEDATA_CLOUDAPPVEHICLEID	
VEHICLEDATA_OEM_CUSTOM_DATA	
VEHICLEDATA_STABILITYCONTROLSSTATUS	
VEHICLEDATA_WINDOWSTATUS	
VEHICLEDATA_HANDSOFFSTEERING	
VEHICLEDATA_SEATOCCUPANCY	

HybridAppPreference

Enumeration for the user's preference of which app type to use when both are available

ELEMENTS

VALUE	DESCRIPTION
MOBILE	
CLOUD	
BOTH	

AppCapabilityType

Enumerations of all available app capability types

ELEMENTS

VALUE	DESCRIPTION
VIDEO_STREAMING	

CapacityUnit

ELEMENTS

VALUE	DESCRIPTION
LITERS	
KILOWATTHOURS	
KILOGRAMS	

ButtonName

Defines the hard (physical) and soft (touchscreen) buttons available from the module

ELEMENTS

VALUE	DESCRIPTION
OK	
PLAY_PAUSE	The button name for the physical Play/Pause toggle that can be used by media apps.
SEEKLEFT	
SEEKRIGHT	
TUNEUP	
TUNEDOWN	
PRESET_0	
PRESET_1	
PRESET_2	
PRESET_3	
PRESET_4	
PRESET_5	
PRESET_6	
PRESET_7	
PRESET_8	

VALUE	DESCRIPTION
PRESET_9	
CUSTOM_BUTTON	
SEARCH	
AC_MAX	Tied to CLIMATE RC modules.
AC	Tied to CLIMATE RC modules.
RECIRCULATE	Tied to CLIMATE RC modules.
FAN_UP	Tied to CLIMATE RC modules.
FAN_DOWN	Tied to CLIMATE RC modules.
TEMP_UP	Tied to CLIMATE RC modules.
TEMP_DOWN	Tied to CLIMATE RC modules.
DEFROST_MAX	Tied to CLIMATE RC modules.
DEFROST	Tied to CLIMATE RC modules.
DEFROST_REAR	Tied to CLIMATE RC modules.
UPPER_VENT	Tied to CLIMATE RC modules.
LOWER_VENT	Tied to CLIMATE RC modules.
VALUE	DESCRIPTION

VALUE	DESCRIPTION
VOLUME_UP	Tied to RADIO RC modules.
VOLUME_DOWN	Tied to RADIO RC modules.
EJECT	Tied to RADIO RC modules.
SOURCE	Tied to RADIO RC modules.
SHUFFLE	Tied to RADIO RC modules.
REPEAT	Tied to RADIO RC modules.
NAV_CENTER_LOCATION	
NAV_ZOOM_IN	
NAV_ZOOM_OUT	
NAV_PAN_UP	
NAV_PAN_UP_RIGHT	
NAV_PAN_RIGHT	
NAV_PAN_DOWN_RIGHT	
NAV_PAN_DOWN	
NAV_PAN_DOWN_LEFT	
VALUE	DESCRIPTION

VALUE	DESCRIPTION
NAV_PAN_LEFT	
NAV_PAN_UP_LEFT	
NAV_TILT_TOGGLE	If supported, this toggles between a top-down view and an angled/3D view. If your app supports different, but substantially similar options, then you may implement those. If you don't implement these or similar options, do not subscribe to this button.
NAV_ROTATE_CLOCKWISE	
NAV_ROTATE_COUNTERCLOCKWISE	
NAV_HEADING_TOGGLE	If supported, this toggles between locking the orientation to north or to the vehicle's heading. If your app supports different, but substantially similar options, then you may implement those. If you don't implement these or similar options, do not subscribe to this button.

MediaClockFormat

ELEMENTS

VALUE	DESCRIPTION
CLOCK1	minutesFieldWidth = 2;minutesFieldMax = 19;secondsFieldWidth = 2;secondsFieldMax = 99;maxHours = 19;maxMinutes = 59;maxSeconds = 59; used for Type II and CID headunits
CLOCK2	minutesFieldWidth = 3;minutesFieldMax = 199;secondsFieldWidth = 2;secondsFieldMax = 99;maxHours = 59;maxMinutes = 59;maxSeconds = 59; used for Type V headunit
CLOCK3	minutesFieldWidth = 2;minutesFieldMax = 59;secondsFieldWidth = 2;secondsFieldMax = 59;maxHours = 9;maxMinutes = 59;maxSeconds = 59; used for GEN1.1 MFD3/4/5 headunits
CLOCKTEXT1	5 characters possible Format: 1
CLOCKTEXT2	5 chars possible Format: 1
CLOCKTEXT3	6 chars possible Format: 1
CLOCKTEXT4	6 chars possible Format: c :

DisplayType

Deprecated since: 5.0.0
See DAES for further infos regarding the displays

ELEMENTS

VALUE	DESCRIPTION
CID	
TYPE2	
TYPE5	
NGN	
GEN2_8_DMA	
GEN2_6_DMA	
MFD3	
MFD4	
MFD5	
GEN3_8-INCH	
SDL_GENERIC	

TextFieldName

ELEMENTS

VALUE	DESCRIPTION
mainField1	The first line of first set of main fields of the persistent display; applies to "Show"
mainField2	The second line of first set of main fields of the persistent display; applies to "Show"
mainField3	The first line of second set of main fields of persistent display; applies to "Show"
mainField4	The second line of second set of main fields of the persistent display; applies to "Show"
statusBar	The status bar on NGN; applies to "Show"
mediaClock	Text value for MediaClock field; applies to "Show"
mediaTrack	The track field of NGN and GEN1.1 MFD displays. This field is only available for media applications; applies to "Show"
templateTitle	The title of the new template that will be displayed; applies to "Show"
alertText1	The first line of the alert text field; applies to "Alert"
alertText2	The second line of the alert text field; applies to "Alert"
alertText3	The third line of the alert text field; applies to "Alert"

VALUE	DESCRIPTION
scrollableMessageBody	Long form body of text that can include newlines and tabs; applies to "ScrollableMessage"
initialInteractionText	First line suggestion for a user response (in the case of VR enabled interaction)
navigationText1	First line of navigation text
navigationText2	Second line of navigation text
ETA	Estimated Time of Arrival time for navigation
totalDistance	Total distance to destination for navigation
audioPassThruDisplayText1	First line of text for audio pass thru
audioPassThruDisplayText2	Second line of text for audio pass thru
sliderHeader	Header text for slider
sliderFooter	Footer text for slider
menuName	Primary text for Choice
secondaryText	Secondary text for Choice
tertiaryText	Tertiary text for Choice
menuTitle	Optional text to label an app menu button (for certain touchscreen platforms).
VALUE	DESCRIPTION

VALUE	DESCRIPTION
locationName	Optional name / title of intended location for SendLocation.
locationDescription	Optional description of intended location / establishment (if applicable) for SendLocation.
addressLines	Optional location address (if applicable) for SendLocation.
phoneNumber	Optional phone number of intended location / establishment (if applicable) for SendLocation.
timeToDestination	Optional time to destination field for ShowConstantTBT
turnText	Turn text for turnList parameter of UpdateTurnList
subtleAlertText1	The first line of the subtle alert text field; applies to SubtleAlert alertText1
subtleAlertText2	The second line of the subtle alert text field; applies to SubtleAlert alertText2
subtleAlertSoftButtonText	A text field in the soft button of a subtle alert; applies to SubtleAlert softButtons
menuCommandSecondaryText	Secondary text for AddCommand
menuCommandTertiaryText	Tertiary text for AddCommand
menuSubMenuSecondaryText	Secondary text for AddSubMenu
VALUE	DESCRIPTION

VALUE	DESCRIPTION
menuSubMenuTertiaryText	Tertiary text for AddSubMenu

ImageFieldName

ELEMENTS

VALUE	DESCRIPTION
softButtonImage	The image field for SoftButton
choiceImage	The first image field for Choice
choiceSecondaryImage	The secondary image field for Choice
vrHelpItem	The image field for vrHelpItem
turnIcon	The image field for Turn
menuIcon	The image field for the menu icon in SetGlobalProperties
cmdIcon	The image field for AddCommand
appIcon	The image field for the app icon (set by setAppIcon)
graphic	The primary image field for Show
secondaryGraphic	The secondary image field for Show
showConstantTBTIcon	The primary image field for ShowConstantTBT
showConstantTBTNextTurnIcon	The secondary image field for ShowConstantTBT
locationImage	The optional image of a destination / location
alertIcon	The image field for Alert
VALUE	DESCRIPTION

VALUE	DESCRIPTION
subMenuIcon	The image field for AddSubMenu.menuIcon
subtleAlertIcon	The image of the subtle alert; applies to SubtleAlert alertIcon
menuCommandSecondaryImage	The secondary image field for AddCommand
menuSubMenuSecondaryImage	The secondary image field for AddSubMenu

CharacterSet

The list of potential character sets

ELEMENTS

VALUE	DESCRIPTION
TYPE2SET	
TYPE5SET	
CID1SET	
CID2SET	
ASCII	ASCII as defined in https://en.wikipedia.org/wiki/ASCII as defined in codes 0-127. Non-printable characters such as tabs and back spaces are ignored.
ISO_8859_1	Latin-1, as defined in https://en.wikipedia.org/wiki/ISO/IEC_8859-1
UTF_8	The UTF-8 character set that uses variable bytes per code point. See https://en.wikipedia.org/wiki/UTF-8 for more details. This is the preferred character set.

TextAlignment

The list of possible alignments, left, right, or centered

ELEMENTS

VALUE	DESCRIPTION
LEFT_ALIGNED	
RIGHT_ALIGNED	
CENTERED	

TBTState

Enumeration that describes possible states of turn-by-turn client or SmartDeviceLink app.

ELEMENTS

VALUE	DESCRIPTION
ROUTE_UPDATE_REQUEST	
ROUTE_ACCEPTED	
ROUTE_REFUSED	
ROUTE_CANCELLED	
ETA_REQUEST	
NEXT_TURN_REQUEST	
ROUTE_STATUS_REQUEST	
ROUTE_SUMMARY_REQUEST	
TRIP_STATUS_REQUEST	
ROUTE_UPDATE_REQUEST_TIMEOUT	

DriverDistractionState

Enumeration that describes possible states of driver distraction.

ELEMENTS

VALUE	DESCRIPTION
DD_ON	
DD_OFF	

ImageType

Contains information about the type of image.

ELEMENTS

VALUE	DESCRIPTION
STATIC	
DYNAMIC	

DeliveryMode

The mode in which the SendLocation request is sent

ELEMENTS

VALUE	DESCRIPTION
PROMPT	
DESTINATION	
QUEUE	

VideoStreamingProtocol

Enum for each type of video streaming protocol type.

ELEMENTS

VALUE	DESCRIPTION
RAW	Raw stream bytes that contains no timestamp data and is the lowest supported video streaming
RTP	RTP facilitates the transfer of real-time data. Information provided by this protocol include timestamps (for synchronization), sequence numbers (for packet loss and reordering detection) and the payload format which indicates the encoded format of the data.
RTSP	The transmission of streaming data itself is not a task of RTSP. Most RTSP servers use the Real-time Transport Protocol (RTP) in conjunction with Real-time Control Protocol (RTCP) for media stream delivery. However, some vendors implement proprietary transport protocols.
RTMP	Real-Time Messaging Protocol (RTMP) was initially a proprietary protocol developed by Macromedia for streaming audio, video and data over the Internet, between a Flash player and a server. Macromedia is now owned by Adobe, which has released an incomplete version of the specification of the protocol for public use.
WEBM	The WebM container is based on a profile of Matroska. WebM initially supported VP8 video and Vorbis audio streams. In 2013 it was updated to accommodate VP9 video and Opus audio.

VideoStreamingCodec

Enum for each type of video streaming codec.

ELEMENTS

VALUE	DESCRIPTION
H264	A block-oriented motion-compensation-based video compression standard. As of 2014 it is one of the most commonly used formats for the recording, compression, and distribution of video content.
H265	High Efficiency Video Coding (HEVC), also known as H.265 and MPEG-H Part 2, is a video compression standard, one of several potential successors to the widely used AVC (H.264 or MPEG-4 Part 10). In comparison to AVC, HEVC offers about double the data compression ratio at the same level of video quality, or substantially improved video quality at the same bit rate. It supports resolutions up to 8192x4320, including 8K UHD.
Theora	Theora is derived from the formerly proprietary VP3 codec, released into the public domain by On2 Technologies. It is broadly comparable in design and bitrate efficiency to MPEG-4 Part 2, early versions of Windows Media Video, and RealVideo while lacking some of the features present in some of these other codecs. It is comparable in open standards philosophy to the BBC's Dirac codec.
VP8	VP8 can be multiplexed into the Matroska-based container format WebM along with Vorbis and Opus audio. The image format WebP is based on VP8's intra-frame coding. VP8's direct successor, VP9, and the emerging royalty-free internet video format AV1 from the Alliance for Open Media (AOMedia) are based on VP8.
VALUE	DESCRIPTION

VALUE	DESCRIPTION
VP9	Similar to VP8, but VP9 is customized for video resolutions beyond high-definition video (UHD) and also enables lossless compression.

AudioStreamingIndicator

ELEMENTS

VALUE	DESCRIPTION
PLAY_PAUSE	Default playback indicator. By default the playback indicator should be PLAY_PAUSE when: - the media app is newly registered on the head unit (after RegisterAppInterface) - the media app was closed by the user (App enters HMI_NONE) - the app sends SetMediaClockTimer with audioStreamingIndicator not set to any value
PLAY	Indicates that a button press of the Play/Pause button starts the audio playback.
PAUSE	Indicates that a button press of the Play/Pause button pauses the current audio playback.
STOP	Indicates that a button press of the Play/Pause button stops the current audio playback.

GlobalProperty

The different global properties.

VALUE	DESCRIPTION
USER_LOCATION	Location of the user's seat of setGlobalProperties
HELPPROMPT	The property helpPrompt of setGlobalProperties
TIMEOUTPROMPT	The property timeoutPrompt of setGlobalProperties
VRHELPTITLE	The property vrHelpTitle of setGlobalProperties
VRHELPITEMS	The property array of vrHelp of setGlobalProperties
MENUNAME	The property in-app menu name of setGlobalProperties
MENUICON	The property in-app menu icon of setGlobalProperties
KEYBOARDPROPERTIES	The on-screen keyboard configuration of setGlobalProperties

CompassDirection

The list of potential compass directions

ELEMENTS

VALUE	DESCRIPTION
NORTH	
NORTHWEST	
WEST	
SOUTHWEST	
SOUTH	
SOUTHEAST	
EAST	
NORTHEAST	

Dimension

The supported dimensions of the GPS

ELEMENTS

VALUE	DESCRIPTION
NO_FIX	No GPS at all
2D	Longitude and latitude
3D	Longitude and latitude and altitude

PRNDL

The selected gear.

ELEMENTS

VALUE	DESCRIPTION
PARK	Parking
REVERSE	Reverse gear
NEUTRAL	No gear
DRIVE	Regular Drive mode
SPORT	Drive Sport mode
LOWGEAR	1st gear hold
FIRST	
SECOND	
THIRD	
FOURTH	
FIFTH	
SIXTH	
SEVENTH	
EIGHTH	
NINTH	
VALUE	DESCRIPTION

VALUE	DESCRIPTION
TENTH	
UNKNOWN	
FAULT	

TransmissionType

Type of transmission used in the vehicle.

ELEMENTS

VALUE	DESCRIPTION
MANUAL	Manual transmission.
AUTOMATIC	Automatic transmission.
SEMI_AUTOMATIC	Semi automatic transmission.
DUAL_CLUTCH	Dual clutch transmission.
CONTINUOUSLY_VARIABLE	Continuously variable transmission(CVT).
INFINITELY_VARIABLE	Infinitely variable transmission.
ELECTRIC_VARIABLE	Electric variable transmission.
DIRECT_DRIVE	Direct drive between engine and wheels.

ComponentVolumeStatus

The volume status of a vehicle component.

VALUE	DESCRIPTION
UNKNOWN	
NORMAL	
LOW	
FAULT	
ALERT	
NOT_SUPPORTED	

TPMS

ELEMENTS

VALUE	DESCRIPTION
UNKNOWN	If set the status of the tire is not known.
SYSTEM_FAULT	TPMS does not function.
SENSOR_FAULT	The sensor of the tire does not function.
LOW	TPMS is reporting a low tire pressure for the tire.
SYSTEM_ACTIVE	TPMS is active and the tire pressure is monitored.
TRAIN	TPMS is reporting that the tire must be trained.
TRAINING_COMPLETE	TPMS reports the training for the tire is completed.
NOT_TRAINED	TPMS reports the tire is not trained.

FuelType

ELEMENTS

VALUE	DESCRIPTION
GASOLINE	
DIESEL	
CNG	For vehicles using compressed natural gas.
LPG	For vehicles using liquefied petroleum gas.
HYDROGEN	For FCEV (fuel cell electric vehicle).
BATTERY	For BEV (Battery Electric Vehicle), PHEV (Plug-in Hybrid Electric Vehicle), solar vehicles and other vehicles which run on a battery.

ElectronicParkBrakeStatus

ELEMENTS

VALUE	DESCRIPTION
CLOSED	Park brake actuators have been fully applied.
TRANSITION	Park brake actuators are transitioning to either Apply/Closed or Release/Open state.
OPEN	Park brake actuators are released.
DRIVE_ACTIVE	When driver pulls the Electronic Park Brake switch while driving "at speed".
FAULT	When system has a fault or is under maintenance.

WarningLightStatus

Reflects the status of a cluster instrument warning light.

ELEMENTS

VALUE	DESCRIPTION
OFF	
ON	
FLASH	
NOT_USED	

VehicleDataNotificationStatus

Reflects the status of a vehicle data notification.

ELEMENTS

VALUE	DESCRIPTION
NOT_SUPPORTED	
NORMAL	
ACTIVE	
NOT_USED	

IgnitionStableStatus

Reflects the ignition switch stability.

ELEMENTS

VALUE	DESCRIPTION
IGNITION_SWITCH_NOT_STABLE	
IGNITION_SWITCH_STABLE	
MISSING_FROM_TRANSMITTER	

IgnitionStatus

Reflects the status of ignition.

VALUE	DESCRIPTION
UNKNOWN	
OFF	
ACCESSORY	
RUN	
START	
INVALID	

VehicleDataEventStatus

Reflects the status of a vehicle data event; e.g. a seat belt event status.

ELEMENTS

VALUE	DESCRIPTION
NO_EVENT	
NO	
YES	
NOT_SUPPORTED	
FAULT	

DeviceLevelStatus

Reflects the reported battery status of the connected device, if reported.

ELEMENTS

VALUE	DESCRIPTION
ZERO_LEVEL_BARS	
ONE_LEVEL_BARS	
TWO_LEVEL_BARS	
THREE_LEVEL_BARS	
FOUR_LEVEL_BARS	
NOT_PROVIDED	

PrimaryAudioSource

Reflects the current primary audio source (if selected).

ELEMENTS

VALUE	DESCRIPTION
NO_SOURCE_SELECTED	
CD	
USB	
USB2	
BLUETOOTH_STEREO_BTST	
LINE_IN	
IPOD	
MOBILE_APP	
AM	
FM	
XM	
DAB	

WiperStatus

Reflects the status of the wipers.

ELEMENTS

VALUE	DESCRIPTION
OFF	
AUTO_OFF	
OFF_MOVING	
MAN_INT_OFF	
MAN_INT_ON	
MAN_LOW	
MAN_HIGH	
MAN_FLICK	
WASH	
AUTO_LOW	
AUTO_HIGH	
COURTESYWIPE	
AUTO_ADJUST	
STALLED	
NO_DATA_EXISTS	
VALUE	DESCRIPTION

VehicleDataStatus

Reflects the status of a binary vehicle data item.

ELEMENTS

VALUE	DESCRIPTION
NO_DATA_EXISTS	
OFF	
ON	

MaintenanceModeStatus

Reflects the status of a vehicle maintenance mode.

ELEMENTS

VALUE	DESCRIPTION
NORMAL	
NEAR	
ACTIVE	
FEATURE_NOT_PRESENT	

VehicleDataActiveStatus

Reflects the status of given vehicle component.

VALUE	DESCRIPTION
`INACTIVE_NOT_CONFIRMED`	
`INACTIVE_CONFIRMED`	
`ACTIVE_NOT_CONFIRMED`	
`ACTIVE_CONFIRMED`	
`FAULT`	

AmbientLightStatus

Reflects the status of the ambient light sensor.

ELEMENTS

VALUE	DESCRIPTION
NIGHT	
TWILIGHT_1	
TWILIGHT_2	
TWILIGHT_3	
TWILIGHT_4	
DAY	
UNKNOWN	
INVALID	

ModuleType

VALUE	DESCRIPTION
CLIMATE	
RADIO	
SEAT	
AUDIO	
LIGHT	
HMI_SETTINGS	

DefrostZone

ELEMENTS

VALUE	DESCRIPTION
FRONT	
REAR	
ALL	
NONE	

VentilationMode

VALUE	DESCRIPTION
UPPER	
LOWER	
BOTH	
NONE	

RadioBand

ELEMENTS

VALUE	DESCRIPTION
AM	
FM	
XM	

RadioState

ELEMENTS

VALUE	DESCRIPTION
ACQUIRING	
ACQUIRED	
MULTICAST	
NOT_FOUND	

TemperatureUnit

ELEMENTS

VALUE	DESCRIPTION
FAHRENHEIT	
CELSIUS	

DoorStatusType

ELEMENTS

VALUE	DESCRIPTION
CLOSED	
LOCKED	
AJAR	
REMOVED	

FileType

Enumeration listing possible file types.

ELEMENTS

VALUE	DESCRIPTION
GRAPHIC_BMP	
GRAPHIC_JPEG	
GRAPHIC_PNG	
AUDIO_WAVE	
AUDIO_MP3	
AUDIO_AAC	
BINARY	
JSON	

FuelCutoffStatus

Reflects the status of the RCM fuel cutoff.

ELEMENTS

VALUE	DESCRIPTION
TERMINATE_FUEL	
NORMAL_OPERATION	
FAULT	

EmergencyEventType

Reflects the emergency event status of the vehicle.

ELEMENTS

VALUE	DESCRIPTION
NO_EVENT	
FRONTAL	
SIDE	
REAR	
ROLLOVER	
NOT_SUPPORTED	
FAULT	

ECallConfirmationStatus

Reflects the status of the eCall Notification.

ELEMENTS

VALUE	DESCRIPTION
NORMAL	
CALL_IN_PROGRESS	
CALL_CANCELLED	
CALL_COMPLETED	
CALL_UNSUCCESSFUL	
ECALL_CONFIGURED_OFF	
CALL_COMPLETE_DTMF_TIMEOUT	

PowerModeQualificationStatus

Reflects the status of the current power mode qualification.

ELEMENTS

VALUE	DESCRIPTION
POWER_MODE_UNDEFINED	
POWER_MODE_EVALUATION_IN_PROGRESS	
NOT_DEFINED	
POWER_MODE_OK	

PowerModeStatus

Reflects the status of the current power mode.

ELEMENTS

VALUE	DESCRIPTION
KEY_OUT	
KEY_RECENTLY_OUT	
KEY_APPROVED_0	
POST_ACCESORY_0	
ACCESORY_1	
POST_IGNITION_1	
IGNITION_ON_2	
RUNNING_2	
CRANK_3	

CarModeStatus

Reflects the status of the current car mode.

ELEMENTS

VALUE	DESCRIPTION
NORMAL	
FACTORY	
TRANSPORT	
CRASH	

VehicleDataResultCode

Enumeration that describes possible result codes of a vehicle data entry request.

ELEMENTS

VALUE	DESCRIPTION
SUCCESS	Individual vehicle data item / DTC / DID request or subscription successful
TRUNCATED_DATA	DTC / DID request successful, however, not all active DTCs or full contents of DID location available
DISALLOWED	This vehicle data item is not allowed for this app by the OEM/Manufactorer of the connected module.
USER_DISALLOWED	The user has not granted access to this type of vehicle data item at this time.
INVALID_ID	The ECU ID referenced is not a valid ID on the bus / system.
VEHICLE_DATA_NOT_AVAILABLE	The requested vehicle data item / DTC / DID is not currently available or responding on the bus / system.
DATA_ALREADY_SUBSCRIBED	The vehicle data item is already subscribed.
DATA_NOT_SUBSCRIBED	The vehicle data item cannot be unsubscribed because it is not currently subscribed.
IGNORED	The request for this item is ignored because it is already in progress.

TurnSignal

Enumeration that describes the status of the turn light indicator.

VALUE	DESCRIPTION
OFF	Turn signal is OFF
LEFT	Left turn signal is on
RIGHT	Right turn signal is on
BOTH	Both signals (left and right) are on.

MenuLayout

How the main menu or submenu is laid out on screen

ELEMENTS

VALUE	DESCRIPTION
LIST	
TILES	

TouchType

ELEMENTS

VALUE	DESCRIPTION
BEGIN	
MOVE	
END	
CANCEL	

PermissionStatus

Enumeration that describes possible permission states of a policy table entry.

ELEMENTS

VALUE	DESCRIPTION
ALLOWED	
DISALLOWED	
USER_DISALLOWED	
USER_CONSENT_PENDING	

KeyboardLayout

Enumeration listing possible keyboard layouts.

ELEMENTS

VALUE	DESCRIPTION
QWERTY	
QWERTZ	
AZERTY	
NUMERIC	

KeyboardEvent

Enumeration listing possible keyboard events.

ELEMENTS

VALUE	DESCRIPTION
KEYPRESS	
ENTRY_SUBMITTED	
ENTRY_VOICE	
ENTRY_CANCELLED	
ENTRY_ABORTED	
INPUT_KEY_MASK_ENABLED	
INPUT_KEY_MASK_DISABLED	

KeypressMode

Enumeration listing possible keyboard events.

ELEMENTS

VALUE	DESCRIPTION
SINGLE_KEYPRESS	Each keypress is individually sent as the user presses the keyboard keys.
QUEUE_KEYPRESSES	The keypresses are queued and a string is eventually sent once the user chooses to submit their entry.
RESEND_CURRENT_ENTRY	The keypresses are queue and a string is sent each time the user presses a keyboard key; the string contains the entire current entry.

KeyboardInputMask

Enumeration listing possible input character masking.

ELEMENTS

VALUE	DESCRIPTION
ENABLE_INPUT_KEY_MASK	
DISABLE_INPUT_KEY_MASK	
USER_CHOICE_INPUT_KEY_MASK	

RequestType

Enumeration listing possible asynchronous requests.

ELEMENTS

VALUE	DESCRIPTION
HTTP	
FILE_RESUME	
AUTH_REQUEST	
AUTH_CHALLENGE	
AUTH_ACK	
PROPRIETARY	
QUERY_APPS	
LAUNCH_APP	
LOCK_SCREEN_ICON_URL	
TRAFFIC_MESSAGE_CHANNEL	
DRIVER_PROFILE	
VOICE_SEARCH	
NAVIGATION	
PHONE	
CLIMATE	
VALUE	DESCRIPTION

VALUE	DESCRIPTION
SETTINGS	
VEHICLE_DIAGNOSTICS	
EMERGENCY	
MEDIA	
FOTA	
OEM_SPECIFIC	
ICON_URL	

AppHMIType

Enumeration listing possible app types.

ELEMENTS

VALUE	DESCRIPTION
DEFAULT	
COMMUNICATION	
MEDIA	
MESSAGING	
NAVIGATION	
INFORMATION	
SOCIAL	
BACKGROUND_PROCESS	
TESTING	
SYSTEM	
PROJECTION	
REMOTE_CONTROL	
WEB_VIEW	

PredefinedLayout

Predefined screen layout.

ELEMENTS

VALUE	DESCRIPTION
DEFAULT	Default media / non-media screen. Can be set as a root screen.
MEDIA	Default Media screen. Can be set as a root screen.
NON-MEDIA	Default Non-media screen. Can be set as a root screen.
ONSCREEN_PRESETS	Custom root media screen containing app-defined onscreen presets. Can be set as a root screen.
NAV_FULLSCREEN_MAP	Custom root template screen containing full screen map with navigation controls. Can be set as a root screen.
NAV_LIST	Custom root template screen containing video represented list. Can be set as a root screen.
NAV_KEYBOARD	Custom root template screen containing video represented keyboard. Can be set as a root screen.
GRAPHIC_WITH_TEXT	Custom root template screen containing half-screen graphic with lines of text. Can be set as a root screen.
TEXT_WITH_GRAPHIC	Custom root template screen containing lines of text with half-screen graphic. Can be set as a root screen.
TILES_ONLY	Custom root template screen containing only tiled SoftButtons. Can be set as a root screen.

VALUE	DESCRIPTION
TEXTBUTTONS_ONLY	Custom root template screen containing only text SoftButtons. Can be set as a root screen.
GRAPHIC_WITH_TILES	Custom root template screen containing half-screen graphic with tiled SoftButtons. Can be set as a root screen.
TILES_WITH_GRAPHIC	Custom root template screen containing tiled SoftButtons with half-screen graphic. Can be set as a root screen.
GRAPHIC_WITH_TEXT_AND_SOFTBUTTONS	Custom root template screen containing half-screen graphic with text and SoftButtons. Can be set as a root screen.
TEXT_AND_SOFTBUTTONS_WITH_GRAPHIC	Custom root template screen containing text and SoftButtons with half-screen graphic. Can be set as a root screen.
GRAPHIC_WITH_TEXTBUTTONS	Custom root template screen containing half-screen graphic with text only SoftButtons. Can be set as a root screen.
TEXTBUTTONS_WITH_GRAPHIC	Custom root template screen containing text only SoftButtons with half-screen graphic. Can be set as a root screen.
LARGE_GRAPHIC_WITH_SOFTBUTTONS	Custom root template screen containing a large graphic and SoftButtons. Can be set as a root screen.
DOUBLE_GRAPHIC_WITH_SOFTBUTTONS	Custom root template screen containing two graphics and SoftButtons. Can be set as a root screen.
VALUE	DESCRIPTION

VALUE	DESCRIPTION
LARGE_GRAPHIC_ONLY	Custom root template screen containing only a large graphic. Can be set as a root screen.
WEB_VIEW	Custom root template allowing in-vehicle WebEngine applications with appropriate permissions to show the application's own web view.

FunctionID

Enumeration linking function names with function IDs in SmartDeviceLink protocol. Assumes enumeration starts at value 0.

ELEMENTS

VALUE	DESCRIPTION
RESERVED	
RegisterAppInterfaceID	
UnregisterAppInterfaceID	
SetGlobalPropertiesID	
ResetGlobalPropertiesID	
AddCommandID	
DeleteCommandID	
AddSubMenuID	
DeleteSubMenuID	
CreateInteractionChoiceSetID	
PerformInteractionID	
DeleteInteractionChoiceSetID	
AlertID	
ShowID	
SpeakID	
VALUE	DESCRIPTION

VALUE	DESCRIPTION
SetMediaClockTimerID	
PerformAudioPassThruID	
EndAudioPassThruID	
SubscribeButtonID	
UnsubscribeButtonID	
SubscribeVehicleDataID	
UnsubscribeVehicleDataID	
GetVehicleDataID	
ReadDIDID	
GetDTCsID	
ScrollableMessageID	
SliderID	
ShowConstantTBTID	
AlertManeuverID	
UpdateTurnListID	

VALUE	DESCRIPTION

VALUE	DESCRIPTION
ChangeRegistrationID	
GenericResponseID	
PutFileID	
DeleteFileID	
ListFilesID	
SetAppIconID	
SetDisplayLayoutID	
DiagnosticMessageID	
SystemRequestID	
SendLocationID	
DialNumberID	
ButtonPressID	
GetInteriorVehicleDataID	
SetInteriorVehicleDataID	
GetWayPointsID	

VALUE	DESCRIPTION

VALUE	DESCRIPTION
SubscribeWayPointsID	
UnsubscribeWayPointsID	
GetSystemCapabilityID	
SendHapticDataID	
SetCloudAppPropertiesID	
GetCloudAppPropertiesID	
PublishAppServiceID	
GetAppServiceDataID	
GetFileID	
PerformAppServiceInteractionID	
UnpublishAppServiceID	
CancelInteractionID	
CloseApplicationID	
ShowAppMenuID	
CreateWindowID	

VALUE	DESCRIPTION

VALUE	DESCRIPTION
DeleteWindowID	
GetInteriorVehicleDataConsentID	
ReleaseInteriorVehicleDataModuleID	
SubtleAlertID	
OnHMIStatusID	
OnAppInterfaceUnregisteredID	
OnButtonEventID	
OnButtonPressID	
OnVehicleDataID	
OnCommandID	
OnTBTClientStateID	
OnDriverDistractionID	
OnPermissionsChangeID	
OnAudioPassThruID	
OnLanguageChangeID	

VALUE	DESCRIPTION

VALUE	DESCRIPTION
OnKeyboardInputID	
OnTouchEventID	
OnSystemRequestID	
OnHashChangeID	
OnInteriorVehicleDataID	
OnWayPointChangeID	
OnRCStatusID	
OnAppServiceDataID	
OnSystemCapabilityUpdatedID	
OnSubtleAlertPressedID	
OnUpdateFileID	
OnUpdateSubMenuID	
OnAppCapabilityUpdatedID	
EncodedSyncPDataID	
SyncPDataID	
VALUE	DESCRIPTION

VALUE	DESCRIPTION
OnEncodedSyncPDataID	
OnSyncPDataID	

messageType

Enumeration linking message types with function types in WiPro protocol. Assumes enumeration starts at value 0. The integer value is used in the protocol binary header.

ELEMENTS

VALUE	DESCRIPTION
request	
response	
notification	

WayPointType

Describes what kind of waypoint is requested/provided.

ELEMENTS

VALUE	DESCRIPTION
ALL	
DESTINATION	

SystemCapabilityType

Enumerations of all available system capability types

ELEMENTS

VALUE	DESCRIPTION
NAVIGATION	
PHONE_CALL	
VIDEO_STREAMING	
REMOTE_CONTROL	
APP_SERVICES	
SEAT_LOCATION	
DISPLAYS	
DRIVER_DISTRACTION	

MassageZone

List possible zones of a multi-contour massage seat.

ELEMENTS

VALUE	DESCRIPTION
LUMBAR	The back of a multi-contour massage seat. or SEAT_BACK
SEAT_CUSHION	The bottom a multi-contour massage seat. or SEAT_BOTTOM

MassageMode

List possible modes of a massage zone.

ELEMENTS

VALUE	DESCRIPTION
OFF	
LOW	
HIGH	

MassageCushion

List possible cushions of a multi-contour massage seat.

VALUE	DESCRIPTION
TOP_LUMBAR	
MIDDLE_LUMBAR	
BOTTOM_LUMBAR	
BACK_BOLSTERS	
SEAT_BOLSTERS	

SeatMemoryActionType

ELEMENTS

VALUE	DESCRIPTION
SAVE	Save current seat positions and settings to seat memory.
RESTORE	Restore / apply the seat memory settings to the current seat.
NONE	No action to be performed.

SupportedSeat

Deprecated since: 6.0.0
List possible seats that is a remote controllable seat.

ELEMENTS

VALUE	DESCRIPTION
DRIVER	
FRONT_PASSENGER	

LightName

Enumeration that describes possible values of light name. The mobile libraries and SDL Core use the name string when referencing these elements.

ELEMENTS

VALUE	DESCRIPTION
FRONT_LEFT_HIGH_BEAM	
FRONT_RIGHT_HIGH_BEAM	
FRONT_LEFT_LOW_BEAM	
FRONT_RIGHT_LOW_BEAM	
FRONT_LEFT_PARKING_LIGHT	
FRONT_RIGHT_PARKING_LIGHT	
FRONT_LEFT_FOG_LIGHT	
FRONT_RIGHT_FOG_LIGHT	
FRONT_LEFT_DAYTIME_RUNNING_LIGHT	
FRONT_RIGHT_DAYTIME_RUNNING_LIGHT	
FRONT_LEFT_TURN_LIGHT	
FRONT_RIGHT_TURN_LIGHT	
REAR_LEFT_FOG_LIGHT	
REAR_RIGHT_FOG_LIGHT	
REAR_LEFT_TAIL_LIGHT	

VALUE	DESCRIPTION
REAR_RIGHT_TAIL_LIGHT	
REAR_LEFT_BRAKE_LIGHT	
REAR_RIGHT_BRAKE_LIGHT	
REAR_LEFT_TURN_LIGHT	
REAR_RIGHT_TURN_LIGHT	
REAR_REGISTRATION_PLATE_LIGHT	
HIGH_BEAMS	Include all high beam lights: front_left and front_right.
LOW_BEAMS	Include all low beam lights: front_left and front_right.
FOG_LIGHTS	Include all fog lights: front_left, front_right, rear_left and rear_right.
RUNNING_LIGHTS	Include all daytime running lights: front_left and front_right.
PARKING_LIGHTS	Include all parking lights: front_left and front_right.
BRAKE_LIGHTS	Include all brake lights: rear_left and rear_right.
REAR_REVERSING_LIGHTS	
SIDE_MARKER_LIGHTS	

VALUE	DESCRIPTION
LEFT_TURN_LIGHTS	Include all left turn signal lights: front_left, rear_left, left_side and mirror_mounted.
RIGHT_TURN_LIGHTS	Include all right turn signal lights: front_right, rear_right, right_side and mirror_mounted.
HAZARD_LIGHTS	Include all hazard lights: front_left, front_right, rear_left and rear_right.
REAR_CARGO_LIGHTS	Cargo lamps illuminate the cargo area.
REAR_TRUCK_BED_LIGHTS	Truck bed lamps light up the bed of the truck.
REAR_TRAILER_LIGHTS	Trailer lights are lamps mounted on a trailer hitch.
LEFT_SPOT_LIGHTS	It is the spotlights mounted on the left side of a vehicle.
RIGHT_SPOT_LIGHTS	It is the spotlights mounted on the right side of a vehicle.
LEFT_PUDDLE_LIGHTS	Puddle lamps illuminate the ground beside the door as the customer is opening or approaching the door.
RIGHT_PUDDLE_LIGHTS	Puddle lamps illuminate the ground beside the door as the customer is opening or approaching the door.
AMBIENT_LIGHTS	
OVERHEAD_LIGHTS	

VALUE	DESCRIPTION
READING_LIGHTS	
TRUNK_LIGHTS	
EXTERIOR_FRONT_LIGHTS	Include exterior lights located in front of the vehicle. For example, fog lights and low beams.
EXTERIOR_REAR_LIGHTS	Include exterior lights located at the back of the vehicle. For example, license plate lights, reverse lights, cargo lights, bed lights and trailer assist lights.
EXTERIOR_LEFT_LIGHTS	Include exterior lights located at the left side of the vehicle. For example, left puddle lights and spot lights.
EXTERIOR_RIGHT_LIGHTS	Include exterior lights located at the right side of the vehicle. For example, right puddle lights and spot lights.
EXTERIOR_ALL_LIGHTS	Include all exterior lights around the vehicle.

LightStatus

ELEMENTS

VALUE	DESCRIPTION

VALUE	DESCRIPTION
ON	
OFF	
RAMP_UP	
RAMP_DOWN	
UNKNOWN	
INVALID	

DisplayMode

ELEMENTS

VALUE	DESCRIPTION
DAY	
NIGHT	
AUTO	

DistanceUnit

ELEMENTS

VALUE	DESCRIPTION
MILES	
KILOMETERS	

MetadataType

ELEMENTS

VALUE	DESCRIPTION

VALUE	DESCRIPTION
mediaTitle	The data in this field contains the title of the currently playing audio track.
mediaArtist	The data in this field contains the artist or creator of the currently playing audio track.
mediaAlbum	The data in this field contains the album title of the currently playing audio track.
mediaYear	The data in this field contains the creation year of the currently playing audio track.
mediaGenre	The data in this field contains the genre of the currently playing audio track.
mediaStation	The data in this field contains the name of the current source for the media.
rating	The data in this field is a rating.
currentTemperature	The data in this field is the current temperature.
maximumTemperature	The data in this field is the maximum temperature for the day.
minimumTemperature	The data in this field is the minimum temperature for the day.
weatherTerm	The data in this field describes the current weather (ex. cloudy, clear, etc.).
humidity	The data in this field describes the current humidity value.

AppServiceType

ELEMENTS

VALUE	DESCRIPTION
MEDIA	
WEATHER	
NAVIGATION	

MediaType

ELEMENTS

VALUE	DESCRIPTION
MUSIC	
PODCAST	
AUDIOBOOK	
OTHER	

NavigationAction

ELEMENTS

VALUE	DESCRIPTION
TURN	Using this action plus a supplied direction can give the type of turn.
EXIT	
STAY	
MERGE	
FERRY	
CAR_SHUTTLE_TRAIN	
WAYPOINT	

NavigationJunction

ELEMENTS

VALUE	DESCRIPTION
REGULAR	A junction that represents a standard intersection with a single road crossing another.
BIFURCATION	A junction where the road splits off into two paths; a fork in the road.
MULTI_CARRIAGEWAY	A junction that has multiple intersections and paths.
ROUNDABOUT	A junction where traffic moves in a single direction around a central, non-traversable point to reach one of the connecting roads.
TRAVERSABLE_ROUNDABOUT	Similar to a roundabout, however the center of the roundabout is fully traversable. Also known as a mini-roundabout.
JUGHANDLE	A junction where lefts diverge to the right, then curve to the left, converting a left turn to a crossing maneuver.
ALL_WAY_YIELD	Multiple way intersection that allows traffic to flow based on priority; most commonly right of way and first in, first out.
TURN_AROUND	A junction designated for traffic turn arounds.

Direction

ELEMENTS

VALUE	DESCRIPTION
LEFT	
RIGHT	

ServiceUpdateReason

ELEMENTS

VALUE	DESCRIPTION
PUBLISHED	The service has just been published with the module and once activated to the primary service of its type, it will be ready for possible consumption.
REMOVED	The service has just been unpublished with the module and is no longer accessible
ACTIVATED	The service is activated as the primary service of this type. All requests dealing with this service type will be handled by this service.
DEACTIVATED	The service has been deactivated as the primary service of its type
MANIFEST_UPDATE	The service has updated its manifest. This could imply updated capabilities

SeekIndicatorType

ELEMENTS

VALUE	DESCRIPTION
TRACK	
TIME	

Structs

AudioPassThruCapabilities

Describes different audio type configurations for PerformAudioPassThru. e.g. {8kHz,8-bit,PCM} The audio is recorded in monaural.

PARAMETERS

VALUE	TYPE	MANDATORY	DESCRIPTION
samplingRate	SamplingRate	True	
bitsPerSample	BitsPerSample	True	
audioType	AudioType	True	

CloudAppProperties

PARAMETERS

VALUE	TYPE	MANDATORY	DESCRIPTION
nicknames	String[]	False	An array of app names a cloud app is allowed to register with. If included in a SetCloudAppPrope rties request, this value will overwrite the existing "nicknames" field in the app policies section of the policy table.
appID	String	True	
enabled	Boolean	False	If true, cloud app will be included in HMI RPC UpdateAppList
authToken	String	False	Used to authenticate websocket connection on app activation
cloudTransport Type	String	False	Specifies the connection type Core should use
hybridAppPrefe rence	HybridAppPreference	False	Specifies the user preference to use the cloud app version or mobile app version when both are available

VALUE	TYPE	MANDATORY	DESCRIPTION
endpoint	String	False	Specifies the endpoint which Core will attempt to connect to when this app is selected

Image

PARAMETERS

VALUE	TYPE	MANDATORY	DESCRIPTION
value	String	True	Either the static hex icon value or the binary image file name identifier (sent by PutFile).
imageType	ImageType	True	Describes, whether it is a static or dynamic image.
isTemplate	Boolean	False	If true, the image is a template image and can be recolored by the HMI

SoftButton

PARAMETERS

VALUE	TYPE	MANDATORY	DESCRIPTION
type	SoftButtonType	True	Describes, whether it is text, highlighted text, icon, or dynamic image. See softButtonType
text	String	False	Optional text to display (if defined as TEXT or BOTH)
image	Image	False	Optional image struct for SoftButton (if defined as IMAGE or BOTH)
isHighlighted	Boolean	False	True, if highlighted False, if not highlighted
softButtonID	Integer	True	Value which is returned via OnButtonPress / OnButtonEvent

VALUE	TYPE	MANDATORY	DESCRIPTION
systemAction	SystemAction	False	Parameter indicating whether selecting a SoftButton shall call a specific system action. This is intended to allow Notifications to bring the callee into full / focus; or in the case of persistent overlays, the overlay can persist when a SoftButton is pressed.

Choice

A choice is an option given to the user, which can be selected either by menu, or through voice recognition system.

PARAMETERS

VALUE	TYPE	MANDATORY	DESCRIPTION
choiceID	Integer	True	
menuName	String	True	
vrCommands	String[]	False	
image	Image	False	
secondaryText	String	False	Optional secondary text to display; e.g. address of POI in a search result entry
tertiaryText	String	False	Optional tertiary text to display; e.g. distance to POI for a search result entry
secondaryImage	Image	False	Optional secondary image struct for choice

VrHelpItem

PARAMETERS

VALUE	TYPE	MANDATORY	DESCRIPTION
text	String	True	Text to display for VR Help item
image	Image	False	Image struct for VR Help item
position	Integer	True	Position to display item in VR Help list

SyncMsgVersion

Specifies the version number of the SmartDeviceLink protocol that is supported by the mobile application

PARAMETERS

VALUE	TYPE	MANDATORY	DESCRIPTION
majorVersion	Integer	True	The major version indicates versions that is not-compatible to previous versions.
minorVersion	Integer	True	The minor version indicates a change to a previous version that should still allow to be run on an older version (with limited functionality)
patchVersion	Integer	False	The patch version indicates a fix to existing functionality in a previous version that should still be able to be run on an older version

FuelRange

PARAMETERS

VALUE	TYPE	MANDATORY	DESCRIPTION
type	FuelType	False	
range	Float	False	The estimate range in KM the vehicle can travel based on fuel level and consumption.
level	Float	False	The relative remaining capacity of this fuel type (percentage).
levelState	ComponentVolumeStatus	False	The fuel level state
capacity	Float	False	The absolute capacity of this fuel type.
capacityUnit	CapacityUnit	False	The unit of the capacity of this fuel type such as liters for gasoline or kWh for batteries.

SingleTireStatus

PARAMETERS

VALUE	TYPE	MANDATORY	DESCRIPTION
status	ComponentVolumeStatus	True	See ComponentVolume Status.
tpms	TPMS	False	The status of TPMS according to the particular tire.
pressure	Float	False	The pressure value of the particular tire in kilo pascal.

BeltStatus

PARAMETERS

VALUE	TYPE	MANDATORY	DESCRIPTION
driverBeltDeployed	VehicleDataEventStatus	True	References signal "VedsDrvBelt_D_Ltchd". See VehicleDataEventStatus.
passengerBeltDeployed	VehicleDataEventStatus	True	References signal "VedsPasBelt_D_Ltchd". See VehicleDataEventStatus.
passengerBuckleBelted	VehicleDataEventStatus	True	References signal "VedsRw1PasBckl_D_Ltchd". See VehicleDataEventStatus.
driverBuckleBelted	VehicleDataEventStatus	True	References signal "VedsRw1DrvBckl_D_Ltchd". See VehicleDataEventStatus.
leftRow2BuckleBelted	VehicleDataEventStatus	True	References signal "VedsRw2lBckl_D_Ltchd". See VehicleDataEventStatus.
passengerChildDetected	VehicleDataEventStatus	True	References signal "VedsRw1PasChld_D_Ltchd". See VehicleDataEventStatus.

VALUE	TYPE	MANDATORY	DESCRIPTION
rightRow2BuckleBelted	VehicleDataEventStatus	True	References signal "VedsRw2rBckl_D_Ltchd". See VehicleDataEventStatus.
middleRow2BuckleBelted	VehicleDataEventStatus	True	References signal "VedsRw2mBckl_D_Ltchd". See VehicleDataEventStatus.
middleRow3BuckleBelted	VehicleDataEventStatus	True	References signal "VedsRw3mBckl_D_Ltchd". See VehicleDataEventStatus.
leftRow3BuckleBelted	VehicleDataEventStatus	True	References signal "VedsRw3lBckl_D_Ltchd". See VehicleDataEventStatus.
rightRow3BuckleBelted	VehicleDataEventStatus	True	References signal "VedsRw3rBckl_D_Ltchd". See VehicleDataEventStatus.
leftRearInflatableBelted	VehicleDataEventStatus	True	References signal "VedsRw2lRib_D_Ltchd". See VehicleDataEventStatus.

VALUE	TYPE	MANDATORY	DESCRIPTION
rightRearInflatableBelted	VehicleDataEventStatus	True	References signal "VedsRw2rRib_D_Ltchd". See VehicleDataEventStatus.
middleRow1BeltDeployed	VehicleDataEventStatus	True	References signal "VedsRw1mBelt_D_Ltchd". See VehicleDataEventStatus.
middleRow1BuckleBelted	VehicleDataEventStatus	True	References signal "VedsRw1mBckl_D_Ltchd". See VehicleDataEventStatus.

Grid

Describes a location (origin coordinates and span) of a vehicle component.

PARAMETERS

VALUE	TYPE	MANDATORY	DESCRIPTION
col	Integer	True	
row	Integer	True	
level	Integer	False	
colspan	Integer	False	
rowspan	Integer	False	
levelspan	Integer	False	

DoorStatus

Describes the status of a parameter of door.

PARAMETERS

VALUE	TYPE	MANDATORY	DESCRIPTION
location	Grid	True	
status	DoorStatusType	True	

GateStatus

Describes the status of a parameter of trunk/hood/etc.

PARAMETERS

VALUE	TYPE	MANDATORY	DESCRIPTION
location	Grid	True	
status	DoorStatusType	True	

WindowState

PARAMETERS

VALUE	TYPE	MANDATORY	DESCRIPTION
approximatePosition	Integer	True	The approximate percentage that the window is open - 0 being fully closed, 100 being fully open
deviation	Integer	True	The percentage deviation of the approximatePosition. e.g. If the approximatePosition is 50 and the deviation is 10, then the window's location is somewhere between 40 and 60.

RoofStatus

Describes the status of a parameter of roof/convertible roof/sunroof/moonroof etc. If roof is open (AJAR), state will determine percentage of roof open.

PARAMETERS

VALUE	TYPE	MANDATORY	DESCRIPTION
location	Grid	True	
status	DoorStatusType	True	
state	WindowState	False	

BodyInformation

PARAMETERS

VALUE	TYPE	MANDATORY	DESCRIPTION
parkBrakeActive	Boolean	True	References signal "PrkBrkActv_B_Actl".
ignitionStableStatus	IgnitionStableStatus	True	References signal "Ignition_Switch_Stable". See IgnitionStableStatus.
ignitionStatus	IgnitionStatus	True	References signal "Ignition_status". See IgnitionStatus.
driverDoorAjar	Boolean	False	References signal "DrStatDrv_B_Actl". Deprecated starting with RPC Spec 7.1.0.
passengerDoorAjar	Boolean	False	References signal "DrStatPsngr_B_Actl". Deprecated starting with RPC Spec 7.1.0.
rearLeftDoorAjar	Boolean	False	References signal "DrStatRl_B_Actl". Deprecated starting with RPC Spec 7.1.0.

VALUE	TYPE	MANDATORY	DESCRIPTION
rearRightDoor Ajar	Boolean	False	References signal "DrStatRr_B_Actl". Deprecated starting with RPC Spec 7.1.0.
doorStatuses	DoorStatus[]	False	Provides status for doors if Ajar/Closed/Locked
gateStatuses	GateStatus[]	False	Provides status for trunk/hood/etc. if Ajar/Closed/Locked
roofStatuses	RoofStatus[]	False	Provides status for roof/convertible roof/sunroof/moonroof etc., if Closed/Ajar/Removed etc.

DeviceStatus

PARAMETERS

VALUE	TYPE	MANDATORY	DESCRIPTION
voiceRecOn	Boolean	True	References signal "CPM_VoiceRec_STAT".
btIconOn	Boolean	True	References signal "BT_ICON".
callActive	Boolean	True	References signal "CPM_Call_Active_STAT".
phoneRoaming	Boolean	True	References signal "CPM_Phone_Roaming_STAT".
textMsgAvailable	Boolean	True	References signal "CPM_TextMsg_AVAL".
battLevelStatus	DeviceLevelStatus	True	Device battery level status. References signal "CPM_Batt_Level_STAT". See DeviceLevelStatus.
stereoAudioOutputMuted	Boolean	True	References signal "CPM_Stereo_Audio_Output".
monoAudioOutputMuted	Boolean	True	References signal "CPM_Mono_Audio_Output".

VALUE	TYPE	MANDATORY	DESCRIPTION
signalLevelStatus	DeviceLevelStatus	True	Device signal level status. References signal "CPM_Signal_Strength_STAT". See DeviceLevelStatus.
primaryAudioSource	PrimaryAudioSource	True	References signal "CPM_Stereo_PAS_Source". See PrimaryAudioSource.
eCallEventActive	Boolean	True	References signal "eCall_Event".

HeadLampStatus

PARAMETERS

VALUE	TYPE	MANDATORY	DESCRIPTION
lowBeamsOn	Boolean	True	Status of the low beam lamps. References signal "HeadLampLoActv_B_Stat".
highBeamsOn	Boolean	True	Status of the high beam lamps. References signal "HeadLghtHiOn_B_Stat".
ambientLightSensorStatus	AmbientLightStatus	False	Status of the ambient light sensor.

AppInfo

Contains detailed information about the registered application.

PARAMETERS

VALUE	TYPE	MANDATORY	DESCRIPTION
appDisplayName	String	True	The name displayed for the mobile application on the mobile device (can differ from the app name set in the initial RAI request).
appBundleID	String	True	The AppBundleID of an iOS application or package name of the Android application. This supports App Launch strategies for each platform.
appVersion	String	True	Represents the build version number of this particular mobile app.
appIcon	String	False	A file reference to the icon utilized by this app (simplifies the process of setting an app icon during app registration).

ECallInfo

PARAMETERS

VALUE	TYPE	MANDATORY	DESCRIPTION
eCallNotificationStatus	VehicleDataNotificationStatus	True	References signal "eCallNotification_4A". See VehicleDataNotificationStatus.
auxECallNotificationStatus	VehicleDataNotificationStatus	True	References signal "eCallNotification". See VehicleDataNotificationStatus.
eCallConfirmationStatus	ECallConfirmationStatus	True	References signal "eCallConfirmation". See ECallConfirmationStatus.

AirbagStatus

PARAMETERS

VALUE	TYPE	MANDATORY	DESCRIPTION
driverAirbagDeployed	VehicleDataEventStatus	True	References signal "VedsDrvBag_D_Ltchd". See VehicleDataEventStatus.
driverSideAirbagDeployed	VehicleDataEventStatus	True	References signal "VedsDrvSideBag_D_Ltchd". See VehicleDataEventStatus.
driverCurtainAirbagDeployed	VehicleDataEventStatus	True	References signal "VedsDrvCrtnBag_D_Ltchd". See VehicleDataEventStatus.
passengerAirbagDeployed	VehicleDataEventStatus	True	References signal "VedsPasBag_D_Ltchd". See VehicleDataEventStatus.
passengerCurtainAirbagDeployed	VehicleDataEventStatus	True	References signal "VedsPasCrtnBag_D_Ltchd". See VehicleDataEventStatus.
driverKneeAirbagDeployed	VehicleDataEventStatus	True	References signal "VedsKneeDrvBag_D_Ltchd". See VehicleDataEventStatus.

VALUE	TYPE	MANDATORY	DESCRIPTION

VALUE	TYPE	MANDATORY	DESCRIPTION
passengerSideAirbagDeployed	VehicleDataEventStatus	True	References signal "VedsPasSideBag_D_Ltchd". See VehicleDataEventStatus.
passengerKneeAirbagDeployed	VehicleDataEventStatus	True	References signal "VedsKneePasBag_D_Ltchd". See VehicleDataEventStatus.

EmergencyEvent

PARAMETERS

VALUE	TYPE	MANDATORY	DESCRIPTION
emergencyEventType	EmergencyEventType	True	References signal "VedsEvntType_D_Ltchd". See EmergencyEventType.
fuelCutoffStatus	FuelCutoffStatus	True	References signal "RCM_FuelCutoff". See FuelCutoffStatus.
rolloverEvent	VehicleDataEventStatus	True	References signal "VedsEvntRoll_D_Ltchd". See VehicleDataEventStatus.
maximumChangeVelocity	Integer	True	References signal "VedsMaxDeltaV_D_Ltchd". Change in velocity in KPH. Additional reserved values: 0x00 No event 0xFE Not supported 0xFF Fault
multipleEvents	VehicleDataEventStatus	True	References signal "VedsMultiEvnt_D_Ltchd". See VehicleDataEventStatus.

ClusterModeStatus

PARAMETERS

VALUE	TYPE	MANDATORY	DESCRIPTION
powerModeActive	Boolean	True	References signal "PowerMode_UB".
powerModeQualificationStatus	PowerModeQualificationStatus	True	References signal "PowerModeQF". See PowerModeQualificationStatus.
carModeStatus	CarModeStatus	True	References signal "CarMode". See CarMode.
powerModeStatus	PowerModeStatus	True	References signal "PowerMode". See PowerMode.

MyKey

PARAMETERS

VALUE	TYPE	MANDATORY	DESCRIPTION
e911Override	VehicleDataStatus	True	Indicates whether e911 override is on. References signal "MyKey_e911Override_St". See VehicleDataStatus.

TireStatus

The status and pressure of the tires.

PARAMETERS

VALUE	TYPE	MANDATORY	DESCRIPTION
pressureTelltale	WarningLightStatus	False	Status of the Tire Pressure Telltale. See WarningLightStatus.
leftFront	SingleTireStatus	False	The status of the left front tire.
rightFront	SingleTireStatus	False	The status of the right front tire.
leftRear	SingleTireStatus	False	The status of the left rear tire.
rightRear	SingleTireStatus	False	The status of the right rear tire.
innerLeftRear	SingleTireStatus	False	The status of the inner left rear.
innerRightRear	SingleTireStatus	False	The status of the inner right rear.

StabilityControlsStatus

PARAMETERS

VALUE	TYPE	MANDATORY	DESCRIPTION
escSystem	VehicleDataStatus	False	true if vehicle stability control is ON, else false
trailerSwayControl	VehicleDataStatus	False	true if vehicle trailer sway control is ON, else false

GPSData

Struct with the GPS data.

PARAMETERS

VALUE	TYPE	MANDATORY	DESCRIPTION
longitudeDegrees	Float	True	
latitudeDegrees	Float	True	
utcYear	Integer	False	The current UTC year.
utcMonth	Integer	False	The current UTC month.
utcDay	Integer	False	The current UTC day.
utcHours	Integer	False	The current UTC hour.
utcMinutes	Integer	False	The current UTC minute.
utcSeconds	Integer	False	The current UTC second.
compassDirection	CompassDirection	False	See CompassDirection.
pdop	Float	False	PDOP. If undefined or unavailable, then value shall be set to 0.

VALUE	TYPE	MANDATORY	DESCRIPTION
hdop	Float	False	HDOP. If value is unknown, value shall be set to 0.
vdop	Float	False	VDOP. If value is unknown, value shall be set to 0.
actual	Boolean	False	True, if actual. False, if inferred.
satellites	Integer	False	Number of satellites in view
dimension	Dimension	False	See Dimension
altitude	Float	False	Altitude in meters
heading	Float	False	The heading. North is 0. Resolution is 0.01
speed	Float	False	The speed in KPH

VALUE	TYPE	MANDATORY	DESCRIPTION
shifted	Boolean	False	True, if GPS lat/long, time, and altitude have been purposefully shifted (requires a proprietary algorithm to unshift). False, if the GPS data is raw and un-shifted. If not provided, then value is assumed False.

VehicleDataResult

Individual published data request result

PARAMETERS

VALUE	TYPE	MANDATORY	DESCRIPTION
dataType	VehicleDataType	True	Defined published data element type.
resultCode	VehicleDataResultCode	True	Published data result code.
oemCustomDataType	String	False	Type of requested oem specific parameter

DIDResult

Individual requested DID result and data

PARAMETERS

VALUE	TYPE	MANDATORY	DESCRIPTION
resultCode	VehicleDataResultCode	True	Individual DID result code.
didLocation	Integer	True	Location of raw data from vehicle data DID
data	String	False	Raw DID-based data returned for requested element.

StartTime

PARAMETERS

VALUE	TYPE	MANDATORY	DESCRIPTION
hours	Integer	True	The hour of the media clock. Some radios only support a max of 19 hours. If out of range, it will be rejected.
minutes	Integer	True	
seconds	Integer	True	

TextField

PARAMETERS

VALUE	TYPE	MANDATORY	DESCRIPTION
name	TextFieldName	True	The name that identifies the field. See TextFieldName.
characterSet	CharacterSet	True	The set of characters that are supported by this text field. All text is sent in UTF-8 format, but not all systems may support all of the characters expressed by UTF-8. All systems will support at least ASCII, but they may support more, either the LATIN-1 character set, or the full UTF-8 character set.
width	Integer	True	The number of characters in one row of this field.
rows	Integer	True	The number of rows of this field.

ImageResolution

PARAMETERS

VALUE	TYPE	MANDATORY	DESCRIPTION
resolutionWidth	Integer	True	The image resolution width.
resolutionHeight	Integer	True	The image resolution height.

ImageField

PARAMETERS

VALUE	TYPE	MANDATORY	DESCRIPTION
name	ImageFieldName	True	The name that identifies the field. See ImageFieldName.
imageTypeSupported	FileType[]	True	The image types that are supported in this field. See FileType.
imageResolution	ImageResolution	False	The image resolution of this field.

TouchCoord

PARAMETERS

VALUE	TYPE	MANDATORY	DESCRIPTION
x	Integer	True	The x coordinate of the touch.
y	Integer	True	The y coordinate of the touch.

TouchEvent

PARAMETERS

VALUE	TYPE	MANDATORY	DESCRIPTION
id	Integer	True	A touch's unique identifier. The application can track the current touch events by id. If a touch event has type begin, the id should be added to the set of touches. If a touch event has type end, the id should be removed from the set of touches.

VALUE	TYPE	MANDATORY	DESCRIPTION
ts	Integer[]	True	The time that the touch was recorded. This number can the time since the beginning of the session or something else as long as the units are in milliseconds. The timestamp is used to determined the rate of change of position of a touch. The application also uses the time to verify whether two touches, with different ids, are part of a single action by the user. If there is only a single timestamp in this array, it is the same for every coordinate in the coordinates array.
c	TouchCoord[]	True	

TouchEventCapabilities

PARAMETERS

VALUE	TYPE	MANDATORY	DESCRIPTION
pressAvailable	Boolean	True	
multiTouchAvailable	Boolean	True	
doublePressAvailable	Boolean	True	

ScreenParams

PARAMETERS

VALUE	TYPE	MANDATORY	DESCRIPTION
resolution	ImageResolution	True	The resolution of the prescribed screen area.
touchEventAvailable	TouchEventCapabilities	False	Types of screen touch events available in screen area.

HMIPermissions

PARAMETERS

VALUE	TYPE	MANDATORY	DESCRIPTION
allowed	HMILevel[]	True	A set of all HMI levels that are permitted for this given RPC.
userDisallowed	HMILevel[]	True	A set of all HMI levels that are prohibited for this given RPC.

ParameterPermissions

PARAMETERS

VALUE	TYPE	MANDATORY	DESCRIPTION
allowed	String[]	True	A set of all parameters that are permitted for this given RPC.
userDisallowed	String[]	True	A set of all parameters that are prohibited for this given RPC.

PermissionItem

PARAMETERS

VALUE	TYPE	MANDATORY	DESCRIPTION
rpcName	String	True	Name of the individual RPC in the policy table.
hmiPermissions	HMIPermissions	True	
parameterPermissions	ParameterPermissions	True	
requireEncryption	Boolean	False	

DisplayCapabilities

Deprecated since: 6.0.0
Contains information about the display capabilities. This struct is deprecated; please see the new SystemCapability DISPLAYS and corresponding struct DisplayCapability

PARAMETERS

VALUE	TYPE	MANDATORY	DESCRIPTION
displayType	DisplayType	True	The type of the display. See DisplayType
displayName	String	False	The name of the display the app is connected to.
textFields	TextField[]	True	A set of all fields that support text data. See TextField
imageFields	ImageField[]	False	A set of all fields that support images. See ImageField
mediaClockFormats	MediaClockFormat[]	True	A set of all supported formats of the media clock. See MediaClockFormat
graphicSupported	Boolean	True	The display's persistent screen supports referencing a static or dynamic image.
templatesAvailable	String[]	False	A set of all predefined persistent display templates available on headunit. To be referenced in SetDisplayLayout.

VALUE	TYPE	MANDATORY	DESCRIPTION
screenParams	ScreenParams	False	A set of all parameters related to a prescribed screen area (e.g. for video / touch input).
numCustomPresetsAvailable	Integer	False	The number of on-screen custom presets available (if any); otherwise omitted.

WindowStatus

Describes the status of a window of a door/liftgate etc.

PARAMETERS

VALUE	TYPE	MANDATORY	DESCRIPTION
location	Grid	True	
state	WindowState	True	

ModuleInfo

Information about an RC module

PARAMETERS

VALUE	TYPE	MANDATORY	DESCRIPTION
moduleId	String	True	UUID of a module. "moduleId + moduleType" uniquely identify a module.
location	Grid	False	Location of a module.
serviceArea	Grid	False	Service area of a module.
allowMultipleAccess	Boolean	False	allow multiple users/apps to access the module or not

ButtonCapabilities

Contains information about a button's capabilities. NOTE: Multiple button capabilities can exist for the same ButtonName with different module ids. The module Id for a button should match the module id for a "primary" module type, such as a climate module or a radio module. Whichever module the id matches the button's id is the module the button will control.

PARAMETERS

VALUE	TYPE	MANDATORY	DESCRIPTION
name	ButtonName	True	The name of the button. See ButtonName.
moduleInfo	ModuleInfo	False	Information about an RC module, including its id.
shortPressAvailable	Boolean	True	The button supports a short press. Whenever the button is pressed short, onButtonPressed(SHORT) will be invoked.
longPressAvailable	Boolean	True	The button supports a LONG press. Whenever the button is pressed long, onButtonPressed(LONG) will be invoked.
upDownAvailable	Boolean	True	The button supports "button down" and "button up". Whenever the button is pressed, onButtonEvent(DOWN) will be invoked. Whenever the button is released, onButtonEvent(UP) will be invoked.

SoftButtonCapabilities

Contains information about a SoftButton's capabilities.

PARAMETERS

VALUE	TYPE	MANDATORY	DESCRIPTION
shortPressAvailable	Boolean	True	The button supports a short press. Whenever the button is pressed short, onButtonPressed(SHORT) will be invoked.
longPressAvailable	Boolean	True	The button supports a LONG press. Whenever the button is pressed long, onButtonPressed(LONG) will be invoked.
upDownAvailable	Boolean	True	The button supports "button down" and "button up". Whenever the button is pressed, onButtonEvent(DOWN) will be invoked. Whenever the button is released, onButtonEvent(UP) will be invoked.
imageSupported	Boolean	True	The button supports referencing a static or dynamic image.

VALUE	TYPE	MANDATORY	DESCRIPTION
textSupported	Boolean	False	The button supports the use of text. If not included, the default value should be considered true that the button will support text.

PresetBankCapabilities

Contains information about on-screen preset capabilities.

PARAMETERS

VALUE	TYPE	MANDATORY	DESCRIPTION
onScreenPresetsAvailable	Boolean	True	Onscreen custom presets are available.

DynamicUpdateCapabilities

PARAMETERS

VALUE	TYPE	MANDATORY	DESCRIPTION
supportedDynamicImageFieldNames	ImageFieldName[]	False	An array of ImageFieldName values for which the system supports sending OnFileUpdate notifications. If you send an Image struct for that image field with a name without having uploaded the image data using PutFile that matches that name, the system will request that you upload the data with PutFile at a later point when the HMI needs it. The HMI will then display the image in the appropriate field. If not sent, assume false.

VALUE	TYPE	MANDATORY	DESCRIPTION
supportsDynamicSubMenus	Boolean	False	If true, the head unit supports dynamic sub-menus by sending OnUpdateSubMenu notifications. If true, you should not send AddCommands that attach to a parentID for an AddSubMenu until OnUpdateSubMenu is received with the menuID. At that point, you should send all AddCommands with a parentID that match the menuID. If not set, assume false.

KeyboardLayoutCapability

Describes the capabilities of a single keyboard layout.

PARAMETERS

VALUE	TYPE	MANDATORY	DESCRIPTION
keyboardLayout	KeyboardLayout	True	
numConfigurableKeys	Integer	True	Number of keys available for special characters, App can customize as per their needs.

KeyboardCapabilities

PARAMETERS

VALUE	TYPE	MANDATORY	DESCRIPTION
maskInputCharactersSupported	Boolean	False	Availability of capability to mask input characters using keyboard. True: Available, False: Not Available
supportedKeyboards	KeyboardLayoutCapability[]	False	Capabilities of supported keyboard layouts by HMI.

WindowCapability

PARAMETERS

VALUE	TYPE	MANDATORY	DESCRIPTION
windowID	Integer	False	The specified ID of the window. This ID is either one used when sending the CreateWindow request, or one of the predefined window ID values from the enum PredefinedWindows. If omitted, value is assumed to be the main window on the main display.
textFields	TextField[]	False	A set of all fields that support text data. See TextField
imageFields	ImageField[]	False	A set of all fields that support images. See ImageField
imageTypeSupported	ImageType[]	False	Provides information about image types supported by the system.
templatesAvailable	String[]	False	A set of all window templates available on the head unit.
VALUE	TYPE	MANDATORY	DESCRIPTION

VALUE	TYPE	MANDATORY	DESCRIPTION
numCustomPresetsAvailable	Integer	False	The number of on-window custom presets available (if any); otherwise omitted.
buttonCapabilities	ButtonCapabilities[]	False	The number of buttons and the capabilities of each on-window button.
softButtonCapabilities	SoftButtonCapabilities[]	False	The number of soft buttons available on-window and the capabilities for each button.
menuLayoutsAvailable	MenuLayout[]	False	An array of available menu layouts. If this parameter is not provided, only the LIST layout is assumed to be available
dynamicUpdateCapabilities	DynamicUpdateCapabilities	False	Contains the head unit's capabilities for dynamic updating features declaring if the module will send dynamic update RPCs.

VALUE	TYPE	MANDATORY	DESCRIPTION
keyboardCapabilities	KeyboardCapabilities	False	See KeyboardCapabilities

WindowTypeCapabilities

PARAMETERS

VALUE	TYPE	MANDATORY	DESCRIPTION
type	WindowType	True	
maximumNumberOfWindows	Integer	True	

DisplayCapability

PARAMETERS

VALUE	TYPE	MANDATORY	DESCRIPTION
displayName	String	False	
windowTypeSupported	WindowTypeCapabilities[]	False	Informs the application how many windows the app is allowed to create per type.

VALUE	TYPE	MANDATORY	DESCRIPTION
windowCapabilities	WindowCapability[]	False	Contains a list of capabilities of all windows related to the app. Once the app has registered the capabilities of all windows are provided. GetSystemCapability still allows requesting window capabilities of all windows. After registration, only windows with capabilities changed will be included. Following cases will cause only affected windows to be included: 1. App creates a new window. After the window is created, a system capability notification will be sent related only to the created window. 2. App sets a new layout to the window. The new layout changes window capabilities. The notification will reflect those changes to the single window.

HMICapabilities

VALUE	TYPE	MANDATORY	DESCRIPTION
navigation	Boolean	False	Availability of build in Nav. True: Available, False: Not Available
phoneCall	Boolean	False	Availability of build in phone. True: Available, False: Not Available
videoStreaming	Boolean	False	Availability of video streaming.
remoteControl	Boolean	False	Availability of remote control feature. True: Available, False: Not Available
appServices	Boolean	False	Availability of App Services functionality. True: Available, False: Not Available
displays	Boolean	False	Availability of displays capability. True: Available, False: Not Available
seatLocation	Boolean	False	Availability of seat location feature. True: Available, False: Not Available

VALUE	TYPE	MANDATORY	DESCRIPTION
driverDistraction	Boolean	False	Availability of driver distraction capability. True: Available, False: Not Available

MenuParams

PARAMETERS

VALUE	TYPE	MANDATORY	DESCRIPTION
parentID	Integer	False	unique ID of the sub menu, the command will be added to. If not provided, it will be provided to the top level of the in application menu.
position	Integer	False	Position within the items that are at top level of the in application menu. 0 will insert at the front. 1 will insert at the second position. if position is greater or equal than the number of items on top level, the sub menu will be appended to the end. If this param was omitted the entry will be added at the end.
menuName	String	True	Text to show in the menu for this sub menu.
secondaryText	String	False	Optional secondary text to display
tertiaryText	String	False	Optional tertiary text to display

TTSChunk

A TTS chunk, that consists of text/phonemes to speak or the name of a file to play, and a TTS type (like text or SAPI)

PARAMETERS

VALUE	TYPE	MANDATORY	DESCRIPTION
text	String	True	The text or phonemes to speak, or the name of the audio file to play. May not be empty.
type	SpeechCapabilities	True	Describes whether the TTS chunk is plain text, a specific phoneme set, or an audio file. See SpeechCapabilities

Turn

PARAMETERS

VALUE	TYPE	MANDATORY	DESCRIPTION
navigationText	String	False	Individual turn text. Must provide at least text or icon for a given turn.
turnIcon	Image	False	Individual turn icon. Must provide at least text or icon for a given turn.

VehicleType

PARAMETERS

VALUE	TYPE	MANDATORY	DESCRIPTION
make	String	False	Make of the vehicle, e.g. Ford
model	String	False	Model of the vehicle, e.g. Fiesta
modelYear	String	False	Model Year of the vehicle, e.g. 2013
trim	String	False	Trim of the vehicle, e.g. SE

KeyboardProperties

Configuration of on-screen keyboard (if available).

PARAMETERS

VALUE	TYPE	MANDATORY	DESCRIPTION
language	Language	False	The keyboard language.
keyboardLayout	KeyboardLayout	False	Desired keyboard layout.
keypressMode	KeypressMode	False	Desired keypress mode. If omitted, this value will be set to RESEND_CURRENT_ENTRY.
limitedCharacterList	String[]	False	Array of keyboard characters to enable. All omitted characters will be greyed out (disabled) on the keyboard. If omitted, the entire keyboard will be enabled.
autoCompleteText	String	False	Deprecated, use autoCompleteList instead.

VALUE	TYPE	MANDATORY	DESCRIPTION
autoComplete List	String[]	False	Allows an app to pre-populate the text field with a list of suggested or completed entries as the user types. If empty, the auto-complete list will be removed from the screen.
maskInputChar acters	KeyboardInputMask	False	Allows an app to mask entered characters on HMI
customKeys	String[]	False	Array of special characters to show in customizable keys. If omitted, keyboard will show default special characters

DeviceInfo

Various information about connecting device.

PARAMETERS

VALUE	TYPE	MANDATORY	DESCRIPTION
hardware	String	False	Device model
firmwareRev	String	False	Device firmware revision
os	String	False	Device OS
osVersion	String	False	Device OS version
carrier	String	False	Device mobile carrier (if applicable)
maxNumberRF COMMPorts	Integer	False	Omitted if connected not via BT.

DateTime

PARAMETERS

VALUE	TYPE	MANDATORY	DESCRIPTION
millisecond	Integer	False	Milliseconds
second	Integer	False	Seconds part of time
minute	Integer	False	Minutes part of time
hour	Integer	False	Hours part of time. Note that this structure accepts time only in 24 Hr format
day	Integer	False	Day of the month
month	Integer	False	Month of the year
year	Integer	False	The year in YYYY format
tz_hour	Integer	False	Time zone offset in Hours wrt UTC.
tz_minute	Integer	False	Time zone offset in Min wrt UTC.

Coordinate

PARAMETERS

VALUE	TYPE	MANDATORY	DESCRIPTION
latitudeDegrees	Float	True	Latitude of the location.
longitudeDegrees	Float	True	Longitude of the location.

OASISAddress

PARAMETERS

VALUE	TYPE	MANDATORY	DESCRIPTION
countryName	String	False	Name of the country (localized)
countryCode	String	False	Name of country (ISO 3166-2)
postalCode	String	False	(PLZ, ZIP, PIN, CAP etc.)
administrative Area	String	False	Portion of country (e.g. state)
subAdministrat iveArea	String	False	Portion of e.g. state (e.g. county)
locality	String	False	Hypernym for e.g. city/village
subLocality	String	False	Hypernym for e.g. district
thoroughfare	String	False	Hypernym for street, road etc.
subThoroughfa re	String	False	Portion of thoroughfare e.g. house number

LocationDetails

PARAMETERS

VALUE	TYPE	MANDATORY	DESCRIPTION
coordinate	Coordinate	False	Latitude/Longitude of the location.
locationName	String	False	Name of location.
addressLines	String[]	False	Location address for display purposes only
locationDescription	String	False	Description intended location / establishment (if applicable)
phoneNumber	String	False	Phone number of location / establishment.
locationImage	Image	False	Image / icon of intended location.
searchAddress	OASISAddress	False	Address to be used by navigation engines for search

NavigationCapability

Extended capabilities for an onboard navigation system

PARAMETERS

VALUE	TYPE	MANDATORY	DESCRIPTION
sendLocationEnabled	Boolean	False	If the module has the ability to add locations to the onboard nav
getWayPointsEnabled	Boolean	False	If the module has the ability to return way points from onboard nav

PhoneCapability

Extended capabilities of the module's phone feature

PARAMETERS

VALUE	TYPE	MANDATORY	DESCRIPTION
dialNumberEnabled	Boolean	False	If the module has the ability to perform dial number

VideoStreamingFormat

Video streaming formats and their specifications.

PARAMETERS

VALUE	TYPE	MANDATORY	DESCRIPTION
protocol	VideoStreamingProtocol	True	Protocol type, see VideoStreamingProtocol
codec	VideoStreamingCodec	True	Codec type, see VideoStreamingCodec

VideoStreamingCapability

Contains information about this system's video streaming capabilities.

PARAMETERS

VALUE	TYPE	MANDATORY	DESCRIPTION
preferredResolution	ImageResolution	False	The preferred resolution of a video stream for decoding and rendering on HMI.
maxBitrate	Integer	False	The maximum bitrate of video stream that is supported, in kbps.
supportedFormats	VideoStreamingFormat[]	False	Detailed information on each format supported by this system, in its preferred order (i.e. the first element in the array is most preferable to the system). Each object will contain a VideoStreamingFormat that describes what can be expected.
hapticSpatialDataSupported	Boolean	False	True if the system can utilize the haptic spatial data from the source being streamed. If not included, it can be assumed the module doesn't support haptic spatial data'.

VALUE	TYPE	MANDATORY	DESCRIPTION
diagonalScreenSize	Float	False	The diagonal screen size in inches.
pixelPerInch	Float	False	PPI is the diagonal resolution in pixels divided by the diagonal screen size in inches.
scale	Float	False	The scaling factor the app should use to change the size of the projecting view.
preferredFPS	Integer	False	The preferred frame rate per second of the head unit. The mobile application / app library may take other factors into account that constrain the frame rate lower than this value, but it should not perform streaming at a higher frame rate than this value.
additionalVideoStreamingCapabilities	VideoStreamingCapability[]	False	

DriverDistractionCapability

PARAMETERS

VALUE	TYPE	MANDATORY	DESCRIPTION
menuLength	Integer	False	The number of items allowed in a Choice Set or Command menu while the driver is distracted
subMenuDepth	Integer	False	The depth of submenus allowed when the driver is distracted. e.g. 3 == top level menu -> submenu -> submenu; 1 == top level menu only

RGBColor

PARAMETERS

VALUE	TYPE	MANDATORY	DESCRIPTION
red	Integer	True	
green	Integer	True	
blue	Integer	True	

TemplateColorScheme

A color scheme for all display layout templates.

PARAMETERS

VALUE	TYPE	MANDATORY	DESCRIPTION
primaryColor	RGBColor	False	The primary "accent" color
secondaryColor	RGBColor	False	The secondary "accent" color
backgroundColor	RGBColor	False	The color of the background

TemplateConfiguration

PARAMETERS

VALUE	TYPE	MANDATORY	DESCRIPTION
template	String	True	Predefined or dynamically created window template. Currently only predefined window template layouts are defined.
dayColorScheme	TemplateColorScheme	False	
nightColorScheme	TemplateColorScheme	False	

SeatLocation

Describes the location of a seat.

PARAMETERS

VALUE	TYPE	MANDATORY	DESCRIPTION
grid	Grid	False	

SeatLocationCapability

Contains information about the locations of each seat

PARAMETERS

VALUE	TYPE	MANDATORY	DESCRIPTION
rows	Integer	False	
columns	Integer	False	
levels	Integer	False	
seats	SeatLocation[]	False	Contains a list of SeatLocation in the vehicle

MassageModeData

Specify the mode of a massage zone.

PARAMETERS

VALUE	TYPE	MANDATORY	DESCRIPTION
massageZone	MassageZone	True	
massageMode	MassageMode	True	

MassageCushionFirmness

The intensity or firmness of a cushion.

PARAMETERS

VALUE	TYPE	MANDATORY	DESCRIPTION
cushion	MassageCushion	True	
firmness	Integer	True	

SeatMemoryAction

PARAMETERS

VALUE	TYPE	MANDATORY	DESCRIPTION
id	Integer	True	
label	String	False	
action	SeatMemoryActionType	True	

SeatControlData

Seat control data corresponds to "SEAT" ModuleType.

PARAMETERS

VALUE	TYPE	MANDATORY	DESCRIPTION
id	SupportedSeat	True	
heatingEnabled	Boolean	False	
coolingEnabled	Boolean	False	
heatingLevel	Integer	False	
coolingLevel	Integer	False	
horizontalPosition	Integer	False	
verticalPosition	Integer	False	
frontVerticalPosition	Integer	False	
backVerticalPosition	Integer	False	
backTiltAngle	Integer	False	
headSupportHorizontalPosition	Integer	False	
headSupportVerticalPosition	Integer	False	

VALUE	TYPE	MANDATORY	DESCRIPTION

VALUE	TYPE	MANDATORY	DESCRIPTION
massageEnabled	Boolean	False	
massageMode	MassageModeData[]	False	
massageCushionFirmness	MassageCushionFirmness[]	False	
memory	SeatMemoryAction	False	

SeatControlCapabilities

PARAMETERS

VALUE	TYPE	MANDATORY	DESCRIPTION
moduleName	String	True	The short friendly name of the seat control module. It should not be used to identify a module by mobile application.
moduleInfo	ModuleInfo	False	Information about an RC module, including its id.
heatingEnabledAvailable	Boolean	False	
coolingEnabledAvailable	Boolean	False	
heatingLevelAvailable	Boolean	False	
coolingLevelAvailable	Boolean	False	
horizontalPositionAvailable	Boolean	False	
verticalPositionAvailable	Boolean	False	
frontVerticalPositionAvailable	Boolean	False	

VALUE	TYPE	MANDATORY	DESCRIPTION
backVerticalPositionAvailable	Boolean	False	
backTiltAngleAvailable	Boolean	False	
headSupportHorizontalPositionAvailable	Boolean	False	
headSupportVerticalPositionAvailable	Boolean	False	
massageEnabledAvailable	Boolean	False	
massageModeAvailable	Boolean	False	
massageCushionFirmnessAvailable	Boolean	False	
memoryAvailable	Boolean	False	

Temperature

PARAMETERS

VALUE	TYPE	MANDATORY	DESCRIPTION
unit	TemperatureUnit	True	Temperature Unit
value	Float	True	Temperature Value in TemperatureUnit specified unit. Range depends on OEM and is not checked by SDL.

RdsData

PARAMETERS

VALUE	TYPE	MANDATORY	DESCRIPTION
PS	String	False	Program Service Name
RT	String	False	Radio Text
CT	String	False	The clock text in UTC format as YYYY-MM-DDThh:mm:ss.sTZD
PI	String	False	Program Identification - the call sign for the radio station
PTY	Integer	False	The program type - The region should be used to differentiate between EU and North America program types
TP	Boolean	False	Traffic Program Identification - Identifies a station that offers traffic
TA	Boolean	False	Traffic Announcement Identification - Indicates an ongoing traffic announcement

VALUE	TYPE	MANDATORY	DESCRIPTION
REG	String	False	Region

StationIDNumber

PARAMETERS

VALUE	TYPE	MANDATORY	DESCRIPTION
countryCode	Integer	False	Binary Representation of ITU Country Code. USA Code is 001.
fccFacilityId	Integer	False	Binary representation of unique facility ID assigned by the FCC; FCC controlled for U.S. territory

SisData

PARAMETERS

VALUE	TYPE	MANDATORY	DESCRIPTION
stationShortName	String	False	Identifies the 4-alpha-character station call sign plus an optional (-FM) extension
stationIDNumber	StationIDNumber	False	Used for network Application. Consists of Country Code and FCC Facility ID.
stationLongName	String	False	Identifies the station call sign or other identifying information in the long format.
stationLocation	GPSData	False	Provides the 3-dimensional geographic station location.
stationMessage	String	False	May be used to convey textual information of general interest to the consumer such as weather forecasts or public service announcements. Includes a high priority delivery feature to convey emergencies that may be in the listening area.

RadioControlData

VALUE	TYPE	MANDATORY	DESCRIPTION
frequencyInteger	Integer	False	The integer part of the frequency ie for 101.7 this value should be 101
frequencyFraction	Integer	False	The fractional part of the frequency for 101.7 is 7
band	RadioBand	False	
rdsData	RdsData	False	
hdRadioEnable	Boolean	False	True if the hd radio is on, false if the radio is off
availableHDs	Integer	False	Number of HD sub-channels if available
availableHdChannels	Integer[]	False	The list of available HD sub-channel indexes. Empty list means no Hd channel is available. Read-only.
hdChannel	Integer	False	Current HD sub-channel if available
signalStrength	Integer	False	

VALUE	TYPE	MANDATORY	DESCRIPTION
signalChangeThreshold	Integer	False	If the signal strength falls below the set value for this parameter, the radio will tune to an alternative frequency
radioEnable	Boolean	False	True if the radio is on, false if the radio is off. If set to false, no other data will be included.
state	RadioState	False	
sisData	SisData	False	Read-only Station Information Service (SIS) data provides basic information about the station such as call sign, as well as information not displayable to the consumer such as the station identification number

ClimateControlData

PARAMETERS

VALUE	TYPE	MANDATORY	DESCRIPTION
fanSpeed	Integer	False	
currentTemperature	Temperature	False	
desiredTemperature	Temperature	False	
acEnable	Boolean	False	
circulateAirEnable	Boolean	False	
autoModeEnable	Boolean	False	
defrostZone	DefrostZone	False	
dualModeEnable	Boolean	False	
acMaxEnable	Boolean	False	
ventilationMode	VentilationMode	False	
heatedSteeringWheelEnable	Boolean	False	value false means disabled/turn off, value true means enabled/turn on.

VALUE	TYPE	MANDATORY	DESCRIPTION
heatedWindshieldEnable	Boolean	False	value false means disabled, value true means enabled.
heatedRearWindowEnable	Boolean	False	value false means disabled, value true means enabled.
heatedMirrorsEnable	Boolean	False	value false means disabled, value true means enabled.
climateEnable	Boolean	False	True if the climate module is on, false if the climate module is off

RadioControlCapabilities

Contains information about a radio control module's capabilities.

PARAMETERS

VALUE	TYPE	MANDATORY	DESCRIPTION
moduleName	String	True	The short friendly name of the radio control module. It should not be used to identify a module by mobile application.
moduleInfo	ModuleInfo	False	Information about an RC module, including its id.
radioEnableAvailable	Boolean	False	Availability of the control of enable/disable radio. True: Available, False: Not Available, Not present: Not Available.
radioBandAvailable	Boolean	False	Availability of the control of radio band. True: Available, False: Not Available, Not present: Not Available.
radioFrequencyAvailable	Boolean	False	Availability of the control of radio frequency. True: Available, False: Not Available, Not present: Not Available.
VALUE	TYPE	MANDATORY	DESCRIPTION

VALUE	TYPE	MANDATORY	DESCRIPTION
hdChannelAvailable	Boolean	False	Availability of the control of HD radio channel. True: Available, False: Not Available, Not present: Not Available.
rdsDataAvailable	Boolean	False	Availability of the getting Radio Data System (RDS) data. True: Available, False: Not Available, Not present: Not Available.
availableHDsAvailable	Boolean	False	Availability of the getting the number of available HD channels. True: Available, False: Not Available, Not present: Not Available.
availableHdChannelsAvailable	Boolean	False	Availability of the list of available HD sub-channel indexes. True: Available, False: Not Available, Not present: Not Available.

VALUE	TYPE	MANDATORY	DESCRIPTION
stateAvailable	Boolean	False	Availability of the getting the Radio state. True: Available, False: Not Available, Not present: Not Available.
signalStrength Available	Boolean	False	Availability of the getting the signal strength. True: Available, False: Not Available, Not present: Not Available.
signalChangeT hresholdAvaila ble	Boolean	False	Availability of the getting the signal Change Threshold. True: Available, False: Not Available, Not present: Not Available.
sisDataAvailab le	Boolean	False	Availability of the getting HD radio Station Information Service (SIS) data. True: Available, False: Not Available, Not present: Not Available.

VALUE	TYPE	MANDATORY	DESCRIPTION
hdRadioEnable Available	Boolean	False	Availability of the control of enable/disable HD radio. True: Available, False: Not Available, Not present: Not Available.
siriusxmRadio Available	Boolean	False	Availability of Sirius XM radio. True: Available, False: Not Available, Not present: Not Available.

ClimateControlCapabilities

Contains information about a climate control module's capabilities.

PARAMETERS

VALUE	TYPE	MANDATORY	DESCRIPTION
moduleName	String	True	The short friendly name of the climate control module. It should not be used to identify a module by mobile application.
moduleInfo	ModuleInfo	False	Information about an RC module, including its id.
currentTemperatureAvailable	Boolean	False	Availability of the reading of current temperature. True: Available, False: Not Available, Not present: Not Available.
fanSpeedAvailable	Boolean	False	Availability of the control of fan speed. True: Available, False: Not Available, Not present: Not Available.
desiredTemperatureAvailable	Boolean	False	Availability of the control of desired temperature. True: Available, False: Not Available, Not present: Not Available.

VALUE	TYPE	MANDATORY	DESCRIPTION
acEnableAvailable	Boolean	False	Availability of the control of turn on/off AC. True: Available, False: Not Available, Not present: Not Available.
acMaxEnableAvailable	Boolean	False	Availability of the control of enable/disable air conditioning is ON on the maximum level. True: Available, False: Not Available, Not present: Not Available.
circulateAirEnableAvailable	Boolean	False	Availability of the control of enable/disable circulate Air mode. True: Available, False: Not Available, Not present: Not Available.
autoModeEnableAvailable	Boolean	False	Availability of the control of enable/disable auto mode. True: Available, False: Not Available, Not present: Not Available.

VALUE	TYPE	MANDATORY	DESCRIPTION
dualModeEnableAvailable	Boolean	False	Availability of the control of enable/disable dual mode. True: Available, False: Not Available, Not present: Not Available.
defrostZoneAvailable	Boolean	False	Availability of the control of defrost zones. True: Available, False: Not Available, Not present: Not Available.
defrostZone	DefrostZone[]	False	A set of all defrost zones that are controllable.
ventilationModeAvailable	Boolean	False	Availability of the control of air ventilation mode. True: Available, False: Not Available, Not present: Not Available.
ventilationMode	VentilationMode[]	False	A set of all ventilation modes that are controllable.

VALUE	TYPE	MANDATORY	DESCRIPTION
heatedSteering WheelAvailable	Boolean	False	Availability of the control (enable/disable) of heated Steering Wheel. True: Available, False: Not Available, Not present: Not Available.
heatedWindshi eldAvailable	Boolean	False	Availability of the control (enable/disable) of heated Windshield. True: Available, False: Not Available, Not present: Not Available.
heatedRearWin dowAvailable	Boolean	False	Availability of the control (enable/disable) of heated Rear Window. True: Available, False: Not Available, Not present: Not Available.

VALUE	TYPE	MANDATORY	DESCRIPTION
heatedMirrors Available	Boolean	False	Availability of the control (enable/disable) of heated Mirrors. True: Available, False: Not Available, Not present: Not Available.
climateEnable Available	Boolean	False	Availability of the control of enable/disable climate control. True: Available, False: Not Available, Not present: Not Available.

EqualizerSettings

Defines the each Equalizer channel settings.

PARAMETERS

VALUE	TYPE	MANDATORY	DESCRIPTION
channelId	Integer	True	
channelName	String	False	read-only channel / frequency name (e.i. "Treble, Midrange, Bass" or "125 Hz")
channelSetting	Integer	True	Reflects the setting, from 0%-100%.

AudioControlData

PARAMETERS

VALUE	TYPE	MANDATORY	DESCRIPTION
source	PrimaryAudioSource	False	In a getter response or a notification, it is the current primary audio source of the system. In a setter request, it is the target audio source that the system shall switch to. If the value is MOBILE_APP, the system shall switch to the mobile media app that issues the setter RPC.
VALUE	TYPE	MANDATORY	DESCRIPTION

VALUE	TYPE	MANDATORY	DESCRIPTION
keepContext	Boolean	False	This parameter shall not be present in any getter responses or notifications. This parameter is optional in a setter request. The default value is false if it is not included. If it is false, the system not only changes the audio source but also brings the default application or system UI associated with the audio source to foreground. If it is true, the system only changes the audio source, but keeps the current application in foreground.
volume	Integer	False	Reflects the volume of audio, from 0%-100%.
equalizerSettings	EqualizerSettings[]	False	Defines the list of supported channels (band) and their current/desired settings on HMI

AudioControlCapabilities

VALUE	TYPE	MANDATORY	DESCRIPTION
moduleName	String	True	The short friendly name of the audio control module. It should not be used to identify a module by mobile application.
moduleInfo	ModuleInfo	False	Information about an RC module, including its id.
sourceAvailable	Boolean	False	Availability of the control of audio source.
keepContextAvailable	Boolean	False	Availability of the keepContext parameter.
volumeAvailable	Boolean	False	Availability of the control of audio volume.
equalizerAvailable	Boolean	False	Availability of the control of Equalizer Settings.
equalizerMaxChannelId	Integer	False	Must be included if equalizerAvailable= true, and assume all IDs starting from 1 to this value are valid

LightCapabilities

PARAMETERS

VALUE	TYPE	MANDATORY	DESCRIPTION
name	LightName	True	
statusAvailable	Boolean	False	Indicates if the status (ON/OFF) can be set remotely. App shall not use read-only values (RAMP_UP/RAMP_DOWN/UNKNOWN/INVALID) in a setInteriorVehicleData request.
densityAvailable	Boolean	False	Indicates if the light's density can be set remotely (similar to a dimmer).
rgbColorSpaceAvailable	Boolean	False	Indicates if the light's color can be set remotely by using the sRGB color space.

LightControlCapabilities

PARAMETERS

VALUE	TYPE	MANDATORY	DESCRIPTION
moduleName	String	True	The short friendly name of the light control module. It should not be used to identify a module by mobile application.
moduleInfo	ModuleInfo	False	Information about an RC module, including its id.
supportedLights	LightCapabilities[]	True	An array of available LightCapabilities that are controllable.

LightState

PARAMETERS

VALUE	TYPE	MANDATORY	DESCRIPTION
id	LightName	True	The name of a light or a group of lights.
status	LightStatus	True	
density	Float	False	
color	RGBColor	False	

LightControlData

VALUE	TYPE	MANDATORY	DESCRIPTION
lightState	LightState[]	True	An array of LightNames and their current or desired status. No change to the status of the LightNames that are not listed in the array.

HMISettingsControlData

Corresponds to "HMI_SETTINGS" ModuleType

PARAMETERS

VALUE	TYPE	MANDATORY	DESCRIPTION
displayMode	DisplayMode	False	
temperatureUnit	TemperatureUnit	False	
distanceUnit	DistanceUnit	False	

HMISettingsControlCapabilities

VALUE	TYPE	MANDATORY	DESCRIPTION
moduleName	String	True	The short friendly name of the hmi setting module. It should not be used to identify a module by mobile application.
moduleInfo	ModuleInfo	False	Information about an RC module, including its id.
distanceUnitAvailable	Boolean	False	Availability of the control of distance unit.
temperatureUnitAvailable	Boolean	False	Availability of the control of temperature unit.
displayModeUnitAvailable	Boolean	False	Availability of the control of HMI display mode.

ModuleData

The moduleType indicates which type of data should be changed and identifies which data object exists in this struct. For example, if the moduleType is CLIMATE then a "climateControlData" should exist

PARAMETERS

VALUE	TYPE	MANDATORY	DESCRIPTION
moduleType	ModuleType	True	
moduleId	String	False	Id of a module, published by System Capability.
radioControlData	RadioControlData	False	
climateControlData	ClimateControlData	False	
seatControlData	SeatControlData	False	
audioControlData	AudioControlData	False	
lightControlData	LightControlData	False	
hmiSettingsControlData	HMISettingsControlData	False	

RemoteControlCapabilities

PARAMETERS

VALUE	TYPE	MANDATORY	DESCRIPTION
climateControl Capabilities	ClimateControlCapabi lities[]	False	If included, the platform supports RC climate controls. For this baseline version, maxsize=1. i.e. only one climate control module is supported.
radioControlCa pabilities	RadioControlCapabilit ies[]	False	If included, the platform supports RC radio controls.
buttonCapabilit ies	ButtonCapabilities[]	False	If included, the platform supports RC button controls with the included button names. NOTE: Multiple button capabilities can exist for the same ButtonName with different module ids. The module Id for a button should match the module id for a "primary" module type, such as a climate module or a radio module. Whichever module the id matches the button's id is the module the button will control.
VALUE	TYPE	MANDATORY	DESCRIPTION

VALUE	TYPE	MANDATORY	DESCRIPTION
audioControlC apabilities	AudioControlCapabiliti es[]	False	If included, the platform supports audio controls.
hmiSettingsCo ntrolCapabilitie s	HMISettingsControlCa pabilities	False	If included, the platform supports hmi setting controls.
lightControlCa pabilities	LightControlCapabiliti es	False	If included, the platform supports light controls.
seatControlCa pabilities	SeatControlCapabiliti es[]	False	If included, the platform supports seat controls.

MetadataTags

PARAMETERS

VALUE	TYPE	MANDATORY	DESCRIPTION
mainField1	MetadataType[]	False	The type of data contained in the "mainField1" text field.
mainField2	MetadataType[]	False	The type of data contained in the "mainField2" text field.
mainField3	MetadataType[]	False	The type of data contained in the "mainField3" text field.
mainField4	MetadataType[]	False	The type of data contained in the "mainField4" text field.

Rectangle

PARAMETERS

VALUE	TYPE	MANDATORY	DESCRIPTION
x	Float	True	The upper left X-coordinate of the rectangle
y	Float	True	The upper left Y-coordinate of the rectangle
width	Float	True	The width of the rectangle
height	Float	True	The height of the rectangle

HapticRect

Defines haptic data for each user control object for video streaming application

PARAMETERS

VALUE	TYPE	MANDATORY	DESCRIPTION
id	Integer	True	A user control spatial identifier
rect	Rectangle	True	The position of the haptic rectangle to be highlighted. The center of this rectangle will be "touched" when a press occurs.

MediaServiceManifest

MediaServiceData

This data is related to what a media service should provide

PARAMETERS

VALUE	TYPE	MANDATORY	DESCRIPTION
mediaType	MediaType	False	The type of the currently playing or paused track.
mediaTitle	String	False	Music: The name of the current track Podcast: The name of the current episode Audiobook: The name of the current chapter
mediaArtist	String	False	Music: The name of the current album artist Podcast: The provider of the podcast (hosts, network, company) Audiobook: The book author's name
mediaAlbum	String	False	Music: The name of the current album Podcast: The name of the current podcast show Audiobook: The name of the current book

VALUE	TYPE	MANDATORY	DESCRIPTION
playlistName	String	False	Music: The name of the playlist or radio station, if the user is playing from a playlist, otherwise, Null Podcast: The name of the playlist, if the user is playing from a playlist, otherwise, Null Audiobook: Likely not applicable, possibly a collection or "playlist" of books
isExplicit	Boolean	False	Whether or not the content currently playing (e.g. the track, episode, or book) contains explicit content
trackPlaybackProgress	Integer	False	Music: The current progress of the track in seconds Podcast: The current progress of the episode in seconds Audiobook: The current progress of the current segment (e.g. the chapter) in seconds

VALUE	TYPE	MANDATORY	DESCRIPTION
trackPlayback Duration	Integer	False	Music: The total duration of the track in seconds Podcast: The total duration of the episode in seconds Audiobook: The total duration of the current segment (e.g. the chapter) in seconds
queuePlayback Progress	Integer	False	Music: The current progress of the playback queue in seconds Podcast: The current progress of the playback queue in seconds Audiobook: The current progress of the playback queue (e.g. the book) in seconds
queuePlayback Duration	Integer	False	Music: The total duration of the playback queue in seconds Podcast: The total duration of the playback queue in seconds Audiobook: The total duration of the playback queue (e.g. the book) in seconds

VALUE	TYPE	MANDATORY	DESCRIPTION
queueCurrentTrackNumber	Integer	False	Music: The current number (1 based) of the track in the playback queue Podcast: The current number (1 based) of the episode in the playback queue Audiobook: The current number (1 based) of the episode in the playback queue (e.g. the chapter number in the book)
queueTotalTrackCount	Integer	False	Music: The total number of tracks in the playback queue Podcast: The total number of episodes in the playback queue Audiobook: The total number of sections in the playback queue (e.g. the number of chapters in the book)

VALUE	TYPE	MANDATORY	DESCRIPTION
mediaImage	Image	False	Music: The album art of the current track Podcast: The podcast or chapter artwork of the current podcast episode Audiobook: The book or chapter artwork of the current audiobook

WeatherServiceManifest

PARAMETERS

VALUE	TYPE	MANDATORY	DESCRIPTION
currentForecastSupported	Boolean	False	
maxMultidayForecastAmount	Integer	False	
maxHourlyForecastAmount	Integer	False	
maxMinutelyForecastAmount	Integer	False	
weatherForLocationSupported	Boolean	False	

WeatherAlert

PARAMETERS

VALUE	TYPE	MANDATORY	DESCRIPTION
title	String	False	
summary	String	False	
expires	DateTime	False	
regions	String[]	False	
severity	String	False	
timeIssued	DateTime	False	

WeatherData

PARAMETERS

VALUE	TYPE	MANDATORY	DESCRIPTION
currentTemperature	Temperature	False	The central temperature depending on the context of the weather data. It could be the present temperature, the temperature of a future minute, the temperature of a future hour, or an average temperature of a future day, for example.
temperatureHigh	Temperature	False	
temperatureLow	Temperature	False	
apparentTemperature	Temperature	False	
apparentTemperatureHigh	Temperature	False	
apparentTemperatureLow	Temperature	False	
weatherSummary	String	False	
VALUE	TYPE	MANDATORY	DESCRIPTION

VALUE	TYPE	MANDATORY	DESCRIPTION
time	DateTime	False	
humidity	Float	False	0 to 1, percentage humidity
cloudCover	Float	False	0 to 1, percentage cloud cover
moonPhase	Float	False	0 to 1, percentage of the moon seen, e.g. 0 = no moon, 0.25 = quarter moon
windBearing	Integer	False	In degrees, true north at 0 degrees
windGust	Float	False	km/hr
windSpeed	Float	False	km/hr
nearestStormBearing	Integer	False	In degrees, true north at 0 degrees
nearestStormDistance	Integer	False	In km
precipAccumulation	Float	False	cm
precipIntensity	Float	False	cm of water per hour

VALUE	TYPE	MANDATORY	DESCRIPTION
precipProbability	Float	False	0 to 1, percentage chance
precipType	String	False	e.g. "rain", "snow", "sleet", "hail"
visibility	Float	False	In km
weatherIcon	Image	False	

WeatherServiceData

This data is related to what a weather service would provide

PARAMETERS

VALUE	TYPE	MANDATORY	DESCRIPTION
location	LocationDetails	True	
currentForecast	WeatherData	False	
minuteForecast	WeatherData[]	False	
hourlyForecast	WeatherData[]	False	
multidayForecast	WeatherData[]	False	
alerts	WeatherAlert[]	False	This array should be ordered with the first object being the current day

NavigationServiceManifest

PARAMETERS

VALUE	TYPE	MANDATORY	DESCRIPTION
acceptsWayPoints	Boolean	False	Informs the subscriber if this service can actually accept way points.

NavigationInstruction

PARAMETERS

VALUE	TYPE	MANDATORY	DESCRIPTION
locationDetails	LocationDetails	True	
action	NavigationAction	True	
eta	DateTime	False	
bearing	Integer	False	The angle at which this instruction takes place. For example, 0 would mean straight, less than 45 is bearing right, greater than 135 is sharp right, between 45 and 135 is a regular right, and 180 is a U-Turn, etc.
junctionType	NavigationJunction	False	
drivingSide	Direction	False	Used to infer which side of the road this instruction takes place. For a U-Turn (action=TURN, bearing=180) this will determine which direction the turn should take place.

VALUE	TYPE	MANDATORY	DESCRIPTION

VALUE	TYPE	MANDATORY	DESCRIPTION
details	String	False	This is a string representation of this instruction, used to display instructions to the users. This is not intended to be read aloud to the users, see the param prompt in NavigationServiceData for that.
image	Image	False	An image representation of this instruction.

NavigationServiceData

This data is related to what a navigation service would provide.

PARAMETERS

VALUE	TYPE	MANDATORY	DESCRIPTION
timeStamp	DateTime	True	This is the timestamp of when the data was generated. This is to ensure any time or distance given in the data can accurately be adjusted if necessary.
origin	LocationDetails	False	
destination	LocationDetails	False	
destinationETA	DateTime	False	
instructions	NavigationInstruction[ ]	False	This array should be ordered with all remaining instructions. The start of this array should always contain the next instruction.
nextInstruction ETA	DateTime	False	

VALUE	TYPE	MANDATORY	DESCRIPTION
nextInstruction Distance	Float	False	The distance to this instruction from current location. This should only be updated ever .1 unit of distance. For more accuracy the consumer can use the GPS location of itself and the next instruction.
nextInstruction DistanceScale	Float	False	Distance till next maneuver (starting from) from previous maneuver.
prompt	String	False	This is a prompt message that should be conveyed to the user through either display or voice (TTS). This param will change often as it should represent the following: approaching instruction, post instruction, alerts that affect the current navigation session, etc.

AppServiceManifest

This manifest contains all the information necessary for the service to be published, activated, and consumers able to interact with it

PARAMETERS

VALUE	TYPE	MANDATORY	DESCRIPTION
serviceName	String	False	Unique name of this service
serviceType	String	True	The type of service that is to be offered by this app. See AppServiceType for known enum equivalent types. Parameter is a string to allow for new service types to be used by apps on older versions of SDL Core.
serviceIcon	Image	False	The icon to be associated with this service. Most likely the same as the appIcon.
allowAppConsumers	Boolean	False	If true, app service consumers beyond the IVI system will be able to access this service. If false, only the IVI system will be able consume the service. If not provided, it is assumed to be false.

VALUE	TYPE	MANDATORY	DESCRIPTION
rpcSpecVersion	SyncMsgVersion	False	This is the max RPC Spec version the app service understands. This is important during the RPC passthrough functionality. If not included, it is assumed the max version of the module is acceptable.
handledRPCs	Integer[]	False	This field contains the Function IDs for the RPCs that this service intends to handle correctly. This means the service will provide meaningful responses.
mediaServiceManifest	MediaServiceManifest	False	
weatherServiceManifest	WeatherServiceManifest	False	
navigationServiceManifest	NavigationServiceManifest	False	

AppServiceRecord

This is the record of an app service publisher that the module has. It should contain the most up to date information including the service's active state

PARAMETERS

VALUE	TYPE	MANDATORY	DESCRIPTION
serviceID	String	True	A unique ID tied to this specific service record. The ID is supplied by the module that services publish themselves.
serviceManifest	AppServiceManifest	True	Manifest for the service that this record is for.
servicePublished	Boolean	True	If true, the service is published and available. If false, the service has likely just been unpublished, and should be considered unavailable.
serviceActive	Boolean	True	If true, the service is the active primary service of the supplied service type. It will receive all potential RPCs that are passed through to that service type. If false, it is not the primary service of the supplied type. See servicePublished for its availability.

AppServiceData

Contains all the current data of the app service. The serviceType will link to which of the service data objects are included in this object (e.g. if the service type is MEDIA, the mediaServiceData param should be included).

PARAMETERS

VALUE	TYPE	MANDATORY	DESCRIPTION
serviceType	String	True	The type of service that is to be offered by this app. See AppServiceType for known enum equivalent types. Parameter is a string to allow for new service types to be used by apps on older versions of SDL Core.
serviceID	String	True	
mediaServiceData	MediaServiceData	False	
weatherServiceData	WeatherServiceData	False	
navigationServiceData	NavigationServiceData	False	

AppServiceCapability

VALUE	TYPE	MANDATORY	DESCRIPTION
updateReason	ServiceUpdateReason	False	Only included in OnSystemCapabilityUpdated. Update reason for service record.
updatedAppServiceRecord	AppServiceRecord	True	Service record for a specific app service provider

AppServicesCapabilities

Capabilities of app services including what service types are supported and the current state of services.

PARAMETERS

VALUE	TYPE	MANDATORY	DESCRIPTION
appServices	AppServiceCapability[]	False	An array of currently available services. If this is an update to the capability the affected services will include an update reason in that item

SystemCapability

The systemCapabilityType identifies which data object exists in this struct. For example, if the SystemCapability Type is NAVIGATION then a "navigationCapability" should exist

PARAMETERS

VALUE	TYPE	MANDATORY	DESCRIPTION
systemCapabilityType	SystemCapabilityType	True	Used as a descriptor of what data to expect in this struct. The corresponding param to this enum should be included and the only other param included.
navigationCapability	NavigationCapability	False	Describes extended capabilities for onboard navigation system
phoneCapability	PhoneCapability	False	Describes extended capabilities of the module's phone feature
videoStreamingCapability	VideoStreamingCapability	False	Contains information about the module's video streaming capabilities
remoteControlCapability	RemoteControlCapabilities	False	Contains information about the module's remote control feature

VALUE	TYPE	MANDATORY	DESCRIPTION
appServicesCapabilities	AppServicesCapabilities	False	An array of currently available services. If this is an update to the capability the affected services will include an update reason in that item
seatLocationCapability	SeatLocationCapability	False	Contains information about the locations of each seat
displayCapabilities	DisplayCapability[]	False	
driverDistractionCapability	DriverDistractionCapability	False	Describes capabilities when the driver is distracted

ClimateData

PARAMETERS

VALUE	TYPE	MANDATORY	DESCRIPTION
externalTemperature	Temperature	False	The external temperature in degrees celsius
cabinTemperature	Temperature	False	Internal ambient cabin temperature in degrees celsius
atmosphericPressure	Float	False	Current atmospheric pressure in mBar

GearStatus

PARAMETERS

VALUE	TYPE	MANDATORY	DESCRIPTION
userSelectedGear	PRNDL	False	Gear position selected by the user i.e. Park, Drive, Reverse
actualGear	PRNDL	False	Actual Gear in use by the transmission
transmissionType	TransmissionType	False	Tells the transmission type

SeatStatus

Describes the status of a parameter of seat.

PARAMETERS

VALUE	TYPE	MANDATORY	DESCRIPTION
seatLocation	SeatLocation	True	
conditionActive	Boolean	True	

SeatOccupancy

PARAMETERS

VALUE	TYPE	MANDATORY	DESCRIPTION
seatsOccupied	SeatStatus[]	False	Seat status array containing location and whether the seats are occupied.
seatsBelted	SeatStatus[]	False	Seat status array containing location and whether the seats are belted.

SeekStreamingIndicator

The seek next / skip previous subscription buttons' content

PARAMETERS

VALUE	TYPE	MANDATORY	DESCRIPTION
type	SeekIndicatorType	True	
seekTime	Integer	False	If the type is TIME, this number of seconds may be present alongside the skip indicator. It will indicate the number of seconds that the currently playing media will skip forward or backward.

AppCapability

PARAMETERS

VALUE	TYPE	MANDATORY	DESCRIPTION
appCapabilityType	AppCapabilityType	True	Used as a descriptor of what data to expect in this struct. The corresponding param to this enum should be included and the only other param included.
videoStreamingCapability	VideoStreamingCapability	False	Describes supported capabilities for video streaming

Remote Procedure Calls

RegisterAppInterface

Message Type: **request**

Establishes an interface with a mobile application. Before registerAppInterface no other commands will be accepted/executed.

PARAMETERS

VALUE	TYPE	MANDATORY	DESCRIPTION
syncMsgVersion	SyncMsgVersion	True	See SyncMsgVersion
appName	String	True	The mobile application name, e.g. "My SDL App". Needs to be unique over all applications from the same device. May not be empty. May not start with a new line character. May not interfere with any name or synonym of previously registered applications from the same device and any predefined blacklist of words (global commands) Additional applications with the same name from the same device will be rejected. Only characters from char set [@TODO: Create char set (character/hex value) for each ACM and refer to] are supported.

VALUE	TYPE	MANDATORY	DESCRIPTION
ttsName	TTSChunk[]	False	TTS string for VR recognition of the mobile application name, e.g. "My S D L App". Meant to overcome any failing on speech engine in properly pronouncing / understanding app name. Needs to be unique over all applications from the same device. May not be empty. May not start with a new line character. Only characters from char set [@TODO: Create char set (character/hex value) for each ACM and refer to] are supported.

VALUE	TYPE	MANDATORY	DESCRIPTION
ngnMediaScreenAppName	String	False	Provides an abbreviated version of the app name (if needed), that will be displayed on the NGN media screen. If not provided, the appName is used instead (and will be truncated if too long) Only characters from char set [@TODO: Create char set (character/hex value) for each ACM and refer to] are supported.

VALUE	TYPE	MANDATORY	DESCRIPTION
vrSynonyms	String[]	False	Defines an additional voice recognition command. May not interfere with any app name of previously registered applications from the same device and any predefined blacklist of words (global commands) Only characters from char set [@TODO: Create char set (character/hex value) for each ACM and refer to] are supported.
isMediaApplication	Boolean	True	Indicates if the application is a media or a non-media application. Only media applications will be able to stream audio to the module that is audible outside of the BT media source.
VALUE	TYPE	MANDATORY	DESCRIPTION

VALUE	TYPE	MANDATORY	DESCRIPTION
languageDesired	Language	True	See Language Current app's expected VR+TTS language If there is a mismatch with the module, the app will be able to change this registration with changeRegistration prior to app being brought into focus.
hmiDisplayLanguageDesired	Language	True	See Language Current app's expected display language If there is a mismatch with the module, the app will be able to change this registration with changeRegistration prior to app being brought into focus.
appHMIType	AppHMIType[]	False	See AppHMIType List of all applicable app HMI types stating which HMI classifications to be given to the app.

VALUE	TYPE	MANDATORY	DESCRIPTION
hashID	String	False	ID used to uniquely identify current state of all app data that can persist through connection cycles (e.g. ignition cycles). This registered data (commands, submenus, choice sets, etc.) can be reestablished without needing to explicitly reregister each piece. If omitted, then the previous state of an app's commands, etc. will not be restored. When sending hashID, all RegisterAppInterface parameters should still be provided (e.g. ttsName, etc.).
deviceInfo	DeviceInfo	False	See DeviceInfo.
appID	String	True	ID used to validate app with policy table entries
fullAppID	String	False	ID used to validate app with policy table entries

VALUE	TYPE	MANDATORY	DESCRIPTION
appInfo	AppInfo	False	See AppInfo.
dayColorScheme	TemplateColorScheme	False	
nightColorScheme	TemplateColorScheme	False	

RegisterAppInterface

Message Type: **response**

The response to registerAppInterface

PARAMETERS

VALUE	TYPE	MANDATORY	DESCRIPTION
success	Boolean	True	true if successful; false, if failed
resultCode	Result	True	See Result
info	String	False	Provides additional human readable info regarding the result.
syncMsgVersion	SyncMsgVersion	False	See SyncMsgVersion
language	Language	False	The currently active VR+TTS language on the module. See "Language" for options.
hmiDisplayLanguage	Language	False	The currently active display language on the module. See "Language" for options.
displayCapabilities	DisplayCapabilities	False	See DisplayCapabilities. This parameter is deprecated and replaced by SystemCapability using DISPLAYS.

VALUE	TYPE	MANDATORY	DESCRIPTION
buttonCapabilities	ButtonCapabilities[]	False	See ButtonCapabilities. This parameter is deprecated and replaced by SystemCapability using DISPLAYS.
softButtonCapabilities	SoftButtonCapabilities[]	False	If returned, the platform supports on-screen SoftButtons; see SoftButtonCapabilities. This parameter is deprecated and replaced by SystemCapability using DISPLAYS.
presetBankCapabilities	PresetBankCapabilities	False	If returned, the platform supports custom on-screen Presets; see PresetBankCapabilities. This parameter is deprecated and replaced by SystemCapability using DISPLAYS.
hmiZoneCapabilities	HmiZoneCapabilities[]	False	See HmiZoneCapabilities

VALUE	TYPE	MANDATORY	DESCRIPTION
speechCapabilities	SpeechCapabilities[]	False	See SpeechCapabilities
prerecordedSpeech	PrerecordedSpeech[]	False	See PrerecordedSpeech
vrCapabilities	VrCapabilities[]	False	See VrCapabilities
audioPassThruCapabilities	AudioPassThruCapabilities[]	False	See AudioPassThruCapability
pcmStreamCapabilities	AudioPassThruCapabilities	False	See AudioPassThruCapability
vehicleType	VehicleType	False	Specifies the vehicle's type. See VehicleType.
supportedDiagModes	Integer[]	False	Specifies the white-list of supported diagnostic modes (0x00-0xFF) capable for DiagnosticMessage requests. If a mode outside this list is requested, it will be rejected.
hmiCapabilities	HMICapabilities	False	Specifies the HMI's capabilities. See HMICapabilities.

VALUE	TYPE	MANDATORY	DESCRIPTION
sdlVersion	String	False	The SmartDeviceLink version.
systemSoftwareVersion	String	False	The software version of the system that implements the SmartDeviceLink core.
iconResumed	Boolean	False	Existence of apps icon at system. If true, apps icon was resumed at system. If false, apps icon is not resumed at system

UnregisterAppInterface

Message Type: **request**

Closes an interface from a mobile application. After unregisterAppInterface, no commands other than registerAppInterface will be accepted/executed. Will fail, if no registerAppInterface was completed successfully before.

UnregisterAppInterface

Message Type: **response**

PARAMETERS

VALUE	TYPE	MANDATORY	DESCRIPTION
success	Boolean	True	true if successful; false, if failed
resultCode	Result	True	See Result
info	String	False	Provides additional human readable info regarding the result.

CreateWindow

Message Type: **request**

Create a new window on the display with the specified window type.

PARAMETERS

VALUE	TYPE	MANDATORY	DESCRIPTION
windowID	Integer	True	A unique ID to identify the window. The value of '0' will always be the default main window on the main display and should not be used in this context as it will already be created for the app. See PredefinedWindows enum. Creating a window with an ID that is already in use will be rejected with INVALID_ID.
windowName	String	True	The window name to be used by the HMI. The name of the pre-created default window will match the app name. Multiple apps can share the same window name except for the default main window. Creating a window with a name which is already in use by the app will result in DUPLICATE_NAME.

VALUE	TYPE	MANDATORY	DESCRIPTION
type	WindowType	True	The type of the window to be created. Main window or widget.
associatedServiceType	String	False	Allows an app to create a widget related to a specific service type. As an example if a MEDIA app becomes active, this app becomes audible and is allowed to play audio. Actions such as skip or play/pause will be directed to this active media app. In case of widgets, the system can provide a single "media" widget which will act as a placeholder for the active media app. It is only allowed to have one window per service type. This means that a media app can only have a single MEDIA widget. Still the app can create widgets omitting this parameter. Those widgets would be available as app specific widgets that are

			permanently included in the HMI. This parameter is related to widgets only. The default main window, which is pre-created during app registration, will be created based on the HMI types specified in the app registration request.
duplicateUpdatesFromWindowID	Integer	False	Optional parameter. Specify whether the content sent to an existing window should be duplicated to the created window. If there isn't a window with the ID, the request will be rejected with INVALID_DATA.

CreateWindow

Message Type: **response**

PARAMETERS

VALUE	TYPE	MANDATORY	DESCRIPTION
success	Boolean	True	true if successful; false, if failed.
info	String	False	Provides additional human readable info regarding the result.
resultCode	Result	True	See Result

DeleteWindow

Message Type: **request**

Deletes previously created window of the SDL application.

PARAMETERS

VALUE	TYPE	MANDATORY	DESCRIPTION
windowID	Integer	True	A unique ID to identify the window. The value of '0' will always be the default main window on the main display and cannot be deleted. See PredefinedWindows enum.

DeleteWindow

Message Type: **response**

PARAMETERS

VALUE	TYPE	MANDATORY	DESCRIPTION
success	Boolean	True	true if successful; false, if failed.
info	String	False	Provides additional human readable info regarding the result.
resultCode	Result	True	See Result

SetGlobalProperties

Message Type: **request**

Allows setting global properties.

PARAMETERS

VALUE	TYPE	MANDATORY	DESCRIPTION
userLocation	SeatLocation	False	Location of the user's seat. Default is driver's seat location if it is not set yet.
helpPrompt	TTSChunk[]	False	The help prompt. An array of text chunks of type TTSChunk. See TTSChunk. The array must have at least one item.
timeoutPrompt	TTSChunk[]	False	Help text for a wait timeout. An array of text chunks of type TTSChunk. See TTSChunk. The array must have at least one item.
vrHelpTitle	String	False	VR Help Title text. If omitted on supported displays, the default module help title shall be used. If omitted and one or more vrHelp items are provided, the request will be rejected.

VALUE	TYPE	MANDATORY	DESCRIPTION
vrHelp	VrHelpItem[]	False	VR Help Items. If omitted on supported displays, the default SmartDeviceLink VR help / What Can I Say? screen shall be used. If the list of VR Help Items contains nonsequential positions (e.g. [1,2,4]), the RPC shall be rejected. If omitted and a vrHelpTitle is provided, the request will be rejected.
menuTitle	String	False	Optional text to label an app menu button (for certain touchscreen platforms).
menuIcon	Image	False	Optional icon to draw on an app menu button (for certain touchscreen platforms).
keyboardProperties	KeyboardProperties	False	On-screen keyboard configuration (if available).

VALUE	TYPE	MANDATORY	DESCRIPTION
menuLayout	MenuLayout	False	Sets the layout of the main menu screen. If this is sent while a menu is already on-screen, the head unit will change the display to the new layout type.

SetGlobalProperties

Message Type: **response**

PARAMETERS

VALUE	TYPE	MANDATORY	DESCRIPTION
success	Boolean	True	true if successful; false, if failed
resultCode	Result	True	See Result
info	String	False	Provides additional human readable info regarding the result.

ResetGlobalProperties

Message Type: **request**

Allows resetting global properties.

PARAMETERS

VALUE	TYPE	MANDATORY	DESCRIPTION
properties	GlobalProperty[]	True	Contains the names of all global properties (like timeoutPrompt) that should be unset. Resetting means, that they have the same value as at start up (default)

ResetGlobalProperties

Message Type: **response**

PARAMETERS

VALUE	TYPE	MANDATORY	DESCRIPTION
success	Boolean	True	true if successful; false, if failed
resultCode	Result	True	See Result
info	String	False	Provides additional human readable info regarding the result.

AddCommand

Message Type: **request**

Adds a command to the in application menu. Either menuParams or vrCommands must be provided.

PARAMETERS

VALUE	TYPE	MANDATORY	DESCRIPTION
cmdID	Integer	True	unique ID of the command to add.
menuParams	MenuParams	False	Optional sub value containing menu parameters
vrCommands	String[]	False	An array of strings to be used as VR synonyms for this command. If this array is provided, it may not be empty.
cmdIcon	Image	False	Image struct determining whether static or dynamic icon. If omitted on supported displays, no (or the default if applicable) icon shall be displayed.
secondaryImage	Image	False	Optional secondary image struct for menu cell

AddCommand

Message Type: **response**

PARAMETERS

VALUE	TYPE	MANDATORY	DESCRIPTION
success	Boolean	True	true if successful; false, if failed
resultCode	Result	True	See Result
info	String	False	Provides additional human readable info regarding the result.

DeleteCommand

Message Type: **request**

Deletes all commands from the in-application menu with the specified command id.

PARAMETERS

VALUE	TYPE	MANDATORY	DESCRIPTION
cmdID	Integer	True	ID of the command(s) to delete.

DeleteCommand

Message Type: **response**

PARAMETERS

VALUE	TYPE	MANDATORY	DESCRIPTION
success	Boolean	True	true if successful; false, if failed
resultCode	Result	True	See Result
info	String	False	Provides additional human readable info regarding the result.

AddSubMenu

Message Type: **request**

Adds a sub menu to the in-application menu.

PARAMETERS

VALUE	TYPE	MANDATORY	DESCRIPTION
menuID	Integer	True	unique ID of the sub menu to add.
position	Integer	False	Position within the items that are at top level of the in application menu. 0 will insert at the front. 1 will insert at the second position. If position is greater or equal than the number of items on top level, the sub menu will be appended to the end. Position of any submenu will always be located before the return and exit options If this param was omitted the entry will be added at the end.
menuName	String	True	Text to show in the menu for this sub menu.
menuIcon	Image	False	The image field for AddSubMenu
menuLayout	MenuLayout	False	Sets the layout of the submenu screen.

VALUE	TYPE	MANDATORY	DESCRIPTION
parentID	Integer	False	unique ID of the sub menu, the command will be added to. If not provided or 0, it will be provided to the top level of the in application menu.
secondaryText	String	False	Optional secondary text to display
tertiaryText	String	False	Optional tertiary text to display
secondaryImage	Image	False	Optional secondary image struct for sub-menu cell

AddSubMenu

Message Type: **response**

PARAMETERS

VALUE	TYPE	MANDATORY	DESCRIPTION
success	Boolean	True	true if successful; false, if failed
resultCode	Result	True	See Result
info	String	False	Provides additional human readable info regarding the result.

DeleteSubMenu

Message Type: **request**

Deletes a submenu from the in-application menu.

PARAMETERS

VALUE	TYPE	MANDATORY	DESCRIPTION
menuID	Integer	True	The "menuID" of the submenu to delete. (See addSubMenu.menuID)

DeleteSubMenu

Message Type: **response**

PARAMETERS

VALUE	TYPE	MANDATORY	DESCRIPTION
success	Boolean	True	true if successful; false, if failed
resultCode	Result	True	See Result
info	String	False	Provides additional human readable info regarding the result.

ShowAppMenu

Message Type: **request**

Shows the built in menu view

PARAMETERS

VALUE	TYPE	MANDATORY	DESCRIPTION
menuID	Integer	False	If omitted the HMI opens the app's menu. If set to a sub-menu ID the HMI opens the corresponding sub-menu previously added using AddSubMenu.

ShowAppMenu

Message Type: **response**

PARAMETERS

VALUE	TYPE	MANDATORY	DESCRIPTION
success	Boolean	True	true if successful; false, if failed
resultCode	Result	True	See Result
info	String	False	Provides additional human readable info regarding the result.

CreateInteractionChoiceSet

Message Type: **request**

creates interaction choice set to be used later by performInteraction

PARAMETERS

VALUE	TYPE	MANDATORY	DESCRIPTION
interactionChoiceSetID	Integer	True	Unique ID used for this interaction choice set.
choiceSet	Choice[]	True	

CreateInteractionChoiceSet

Message Type: **response**

PARAMETERS

VALUE	TYPE	MANDATORY	DESCRIPTION
success	Boolean	True	true if successful; false, if failed
resultCode	Result	True	See Result
info	String	False	Provides additional human readable info regarding the result.

PerformInteraction

Message Type: **request**

Triggers an interaction (e.g. "Permit GPS?" - Yes, no, Always Allow).

PARAMETERS

VALUE	TYPE	MANDATORY	DESCRIPTION
initialText	String	True	Text to be displayed first.
initialPrompt	TTSChunk[]	False	This is the initial prompt spoken to the user at the start of an interaction. An array of text chunks of type TTSChunk. See TTSChunk. The array must have at least one item.
interactionMode	InteractionMode	True	See InteractionMode.
interactionChoiceSetIDList	Integer[]	True	List of interaction choice set IDs to use with an interaction.
helpPrompt	TTSChunk[]	False	Help text. This is the spoken string when a user speaks "help" when the interaction is occurring. An array of text chunks of type TTSChunk. See TTSChunk. The array must have at least one item.

VALUE	TYPE	MANDATORY	DESCRIPTION
timeoutPrompt	TTSChunk[]	False	Timeout text. This text is spoken when a VR interaction times out. An array of text chunks of type TTSChunk. See TTSChunk. The array must have at least one item.
timeout	Integer	False	Timeout in milliseconds. If omitted a standard value of 10000 milliseconds is used. Applies only to the menu portion of the interaction. The VR timeout will be handled by the platform.
vrHelp	VrHelpItem[]	False	Ability to send suggested VR Help Items to display on-screen during Perform Interaction. If omitted on supported displays, the default generated list of suggested choices shall be displayed.
interactionLayout	LayoutMode	False	See LayoutMode.

VALUE	TYPE	MANDATORY	DESCRIPTION
cancelID	Integer	False	An ID for this specific PerformInteraction to allow cancellation through the CancelInteraction RPC.

PerformInteraction

Message Type: **response**

PARAMETERS

VALUE	TYPE	MANDATORY	DESCRIPTION
success	Boolean	True	true if successful; false, if failed
resultCode	Result	True	See Result
info	String	False	Provides additional human readable info regarding the result.
choiceID	Integer	False	ID of the choice that was selected in response to PerformInteraction. Only is valid if general result is "success:true".
manualTextEntry	String	False	Manually entered text selection, e.g. through keyboard Can be returned in lieu of choiceID, depending on trigger source
triggerSource	TriggerSource	False	See TriggerSource Only is valid if resultCode is SUCCESS.

DeleteInteractionChoiceSet

Message Type: **request**

Deletes interaction choice set that has been created with "CreateInteractionChoiceSet". The interaction may only be deleted when not currently in use by a "performInteraction".

PARAMETERS

VALUE	TYPE	MANDATORY	DESCRIPTION
interactionChoiceSetID	Integer	True	ID of the interaction choice set to delete.

DeleteInteractionChoiceSet

Message Type: **response**

PARAMETERS

VALUE	TYPE	MANDATORY	DESCRIPTION
success	Boolean	True	true if successful; false, if failed
resultCode	Result	True	See Result
info	String	False	Provides additional human readable info regarding the result.

Alert

Message Type: **request**

Shows an alert which typically consists of text-to-speech message and text on the display. At least either alertText1, alertText2 or TTSChunks need to be provided.

PARAMETERS

VALUE	TYPE	MANDATORY	DESCRIPTION
alertText1	String	False	The first line of the alert text field
alertText2	String	False	The second line of the alert text field
alertText3	String	False	The optional third line of the alert text field
ttsChunks	TTSChunk[]	False	An array of text chunks of type TTSChunk. See TTSChunk. The array must have at least one item.
duration	Integer	False	Timeout in milliseconds. Typical timeouts are 3-5 seconds. If omitted, timeout is set to 5s.
playTone	Boolean	False	Defines if tone should be played. Tone is played before TTS. If omitted or provided without ttsChunks, no tone is played.

VALUE	TYPE	MANDATORY	DESCRIPTION
progressIndicator	Boolean	False	If supported on the given platform, the alert GUI will include some sort of animation indicating that loading of a feature is progressing. e.g. a spinning wheel or hourglass, etc.
softButtons	SoftButton[]	False	App defined SoftButtons. If omitted on supported displays, the displayed alert shall not have any SoftButtons.
alertIcon	Image	False	Image struct determining whether static or dynamic icon. If omitted on supported displays, no (or the default if applicable) icon should be displayed.
cancelID	Integer	False	An ID for this specific alert to allow cancellation through the CancelInteraction RPC.

Alert

Message Type: **response**

PARAMETERS

VALUE	TYPE	MANDATORY	DESCRIPTION
success	Boolean	True	true if successful; false, if failed
resultCode	Result	True	See Result
info	String	False	Provides additional human readable info regarding the result.
tryAgainTime	Integer	False	Amount of time (in seconds) that an app must wait before resending an alert. If provided, another system event or overlay currently has a higher priority than this alert. An app must not send an alert without waiting at least the amount of time dictated.

SubtleAlert

Message Type: **request**

Shows an alert which typically consists of text-to-speech message and text on the display.
At least either alertText1, alertText2 or ttsChunks need to be provided.

PARAMETERS

VALUE	TYPE	MANDATORY	DESCRIPTION
alertText1	String	False	The first line of the alert text field
alertText2	String	False	The second line of the alert text field
alertIcon	Image	False	Image to be displayed for the corresponding alert. See Image. If omitted on supported displays, no (or the default if applicable) icon should be displayed.
ttsChunks	TTSChunk[]	False	An array of text chunks of type TTSChunk. See TTSChunk. The array must have at least one item.
duration	Integer	False	Timeout in milliseconds. Typical timeouts are 3-5 seconds. If omitted, timeout is set to 5s.

VALUE	TYPE	MANDATORY	DESCRIPTION
softButtons	SoftButton[]	False	App defined SoftButtons. If omitted on supported displays, the displayed alert shall not have any SoftButtons.
cancelID	Integer	False	An ID for this specific alert to allow cancellation through the CancelInteraction RPC.

SubtleAlert

Message Type: **response**

PARAMETERS

VALUE	TYPE	MANDATORY	DESCRIPTION
success	Boolean	True	true if successful; false, if failed
resultCode	Result	True	See Result
info	String	False	Provides additional human readable info regarding the result.
tryAgainTime	Integer	False	Amount of time (in milliseconds) that an app must wait before resending an alert. If provided, another system event or overlay currently has a higher priority than this alert. An app must not send an alert without waiting at least the amount of time dictated.

OnSubtleAlertPressed

Message Type: **notification**

Sent when the alert itself is touched (outside of a soft button). Touching (or otherwise selecting) the alert should open the app before sending this notification.

Show

Message Type: **request**

Updates the persistent display. Supported fields depend on display capabilities.

PARAMETERS

VALUE	TYPE	MANDATORY	DESCRIPTION
mainField1	String	False	The text that should be displayed in a single or upper display line. If this text is not set, the text of mainField1 stays unchanged. If this text is empty "", the field will be cleared.
mainField2	String	False	The text that should be displayed on the second display line. If this text is not set, the text of mainField2 stays unchanged. If this text is empty "", the field will be cleared.
mainField3	String	False	The text that should be displayed on the second "page" first display line. If this text is not set, the text of mainField3 stays unchanged. If this text is empty "", the field will be cleared.

VALUE	TYPE	MANDATORY	DESCRIPTION
mainField4	String	False	The text that should be displayed on the second "page" second display line. If this text is not set, the text of mainField4 stays unchanged. If this text is empty "", the field will be cleared.
alignment	TextAlignment	False	Specifies how mainField1 and mainField2 texts should be aligned on display. If omitted, texts will be centered.
statusBar	String	False	Requires investigation regarding the nav display capabilities. Potentially lower lowerStatusBar, upperStatusBar, titleBar, etc.

VALUE	TYPE	MANDATORY	DESCRIPTION
mediaClock	String	False	Text value for MediaClock field. Has to be properly formatted by Mobile App according to the module's capabilities. If this text is set, any automatic media clock updates previously set with SetMediaClockTimer will be stopped.
mediaTrack	String	False	The text that should be displayed in the track field. If this text is not set, the text of mediaTrack stays unchanged. If this text is empty "", the field will be cleared.
graphic	Image	False	Image struct determining whether static or dynamic image to display in app. If omitted on supported displays, the displayed graphic shall not change.

VALUE	TYPE	MANDATORY	DESCRIPTION
secondaryGraphic	Image	False	Image struct determining whether static or dynamic secondary image to display in app. If omitted on supported displays, the displayed secondary graphic shall not change.
softButtons	SoftButton[]	False	App defined SoftButtons. If omitted on supported displays, the currently displayed SoftButton values will not change.
customPresets	String[]	False	App labeled on-screen presets (i.e. on-screen media presets or dynamic search suggestions). If omitted on supported displays, the presets will be shown as not defined.

VALUE	TYPE	MANDATORY	DESCRIPTION
metadataTags	MetadataTags	False	App defined metadata information. See MetadataStruct. Uses mainField1, mainField2, mainField3, mainField4. If omitted on supported displays, the currently set metadata tags will not change. If any text field contains no tags or the none tag, the metadata tag for that textfield should be removed.
templateTitle	String	False	The title of the new template that will be displayed. How this will be displayed is dependent on the OEM design and implementation of the template.

VALUE	TYPE	MANDATORY	DESCRIPTION
windowID	Integer	False	This is the unique ID assigned to the window that this RPC is intended. If this param is not included, it will be assumed that this request is specifically for the main window on the main display. See PredefinedWindows enum.
templateConfiguration	TemplateConfiguration	False	Used to set an alternate template layout to a window.

Show

Message Type: **response**

PARAMETERS

VALUE	TYPE	MANDATORY	DESCRIPTION
success	Boolean	True	true if successful; false, if failed
resultCode	Result	True	See Result
info	String	False	Provides additional human readable info regarding the result.

Speak

Message Type: **request**

Speaks a text.

PARAMETERS

VALUE	TYPE	MANDATORY	DESCRIPTION
ttsChunks	TTSChunk[]	True	An array of text chunks of type TTSChunk. See TTSChunk. The array must have at least one item.

Speak

Message Type: **response**

PARAMETERS

VALUE	TYPE	MANDATORY	DESCRIPTION
success	Boolean	True	true if successful; false, if failed
resultCode	Result	True	See Result
info	String	False	Provides additional human readable info regarding the result.

SetMediaClockTimer

Message Type: **request**

Sets the initial media clock value and automatic update method.

PARAMETERS

VALUE	TYPE	MANDATORY	DESCRIPTION
startTime	StartTime	False	See StartTime. startTime must be provided for "COUNTUP" and "COUNTDOWN". startTime will be ignored for "RESUME", and "CLEAR" startTime can be sent for "PAUSE", in which case it will update the paused startTime

VALUE	TYPE	MANDATORY	DESCRIPTION
endTime	StartTime	False	See StartTime. endTime can be provided for "COUNTUP" and "COUNTDOWN"; to be used to calculate any visual progress bar (if not provided, this feature is ignored) If endTime is greater than startTime for COUNTDOWN or less than startTime for COUNTUP, then the request will return an INVALID_DATA. endTime will be ignored for "RESUME", and "CLEAR" endTime can be sent for "PAUSE", in which case it will update the paused endTime

VALUE	TYPE	MANDATORY	DESCRIPTION
updateMode	UpdateMode	True	Enumeration to control the media clock. In case of pause, resume, or clear, the start time value is ignored and shall be left out. For resume, the time continues with the same value as it was when paused.
audioStreamingIndicator	AudioStreamingIndicator	False	Enumeration for the indicator icon on a play/pause button. see AudioStreamingIndicator.
forwardSeekIndicator	SeekStreamingIndicator	False	Used to control the forward seek button to either skip forward a set amount of time or to the next track.
backSeekIndicator	SeekStreamingIndicator	False	Used to control the back seek button to either skip back a set amount of time or to the previous track.

VALUE	TYPE	MANDATORY	DESCRIPTION
countRate	Float	False	The value of this parameter is the amount that the media clock timer will advance per 1.0 seconds of real time. Values less than 1.0 will therefore advance the timer slower than real-time, while values greater than 1.0 will advance the timer faster than real-time. e.g. If this parameter is set to 0.5, the timer will advance one second per two seconds real-time, or at 50% speed. If this parameter is set to 2.0, the timer will advance two seconds per one second real-time, or at 200% speed.

SetMediaClockTimer

Message Type: **response**

PARAMETERS

VALUE	TYPE	MANDATORY	DESCRIPTION
success	Boolean	True	true if successful; false, if failed
resultCode	Result	True	See Result
info	String	False	Provides additional human readable info regarding the result.

PerformAudioPassThru

Message Type: **request**

Starts audio pass thru session

PARAMETERS

VALUE	TYPE	MANDATORY	DESCRIPTION
initialPrompt	TTSChunk[]	False	The module will speak this prompt before opening the audio pass thru session. An array of text chunks of type TTSChunk. See TTSChunk. The array must have at least one item. If omitted, then no initial prompt is spoken.
audioPassThru DisplayText1	String	False	First line of text displayed during audio capture.
audioPassThru DisplayText2	String	False	Second line of text displayed during audio capture.
samplingRate	SamplingRate	True	This value shall be allowed at 8 kHz or 16 or 22 or 44 kHz.
maxDuration	Integer	True	The maximum duration of audio recording in milliseconds.
bitsPerSample	BitsPerSample	True	Specifies the quality the audio is recorded. Currently 8 bit or 16 bit.

VALUE	TYPE	MANDATORY	DESCRIPTION
audioType	AudioType	True	Specifies the type of audio data being requested.
muteAudio	Boolean	False	Defines if the current audio source should be muted during the APT session. If not, the audio source will play without interruption. If omitted, the value is set to true.

PerformAudioPassThru

Message Type: **response**

PARAMETERS

VALUE	TYPE	MANDATORY	DESCRIPTION
success	Boolean	True	true if successful; false, if failed
resultCode	Result	True	See Result
info	String	False	Provides additional human readable info regarding the result.

EndAudioPassThru

Message Type: **request**

When this request is invoked, the audio capture stops.

EndAudioPassThru

Message Type: **response**

PARAMETERS

VALUE	TYPE	MANDATORY	DESCRIPTION
success	Boolean	True	true if successful; false, if failed
resultCode	Result	True	See Result
info	String	False	Provides additional human readable info regarding the result.

SubscribeButton

Message Type: **request**

Subscribes to built-in HMI buttons. The application will be notified by the OnButtonEvent and OnButtonPress. To unsubscribe the notifications, use unsubscribeButton.

PARAMETERS

VALUE	TYPE	MANDATORY	DESCRIPTION
buttonName	ButtonName	True	Name of the button to subscribe.

SubscribeButton

Message Type: **response**

PARAMETERS

VALUE	TYPE	MANDATORY	DESCRIPTION
success	Boolean	True	true if successful; false, if failed
resultCode	Result	True	See Result
info	String	False	Provides additional human readable info regarding the result.

UnsubscribeButton

Message Type: **request**

Unsubscribes from built-in HMI buttons.

PARAMETERS

VALUE	TYPE	MANDATORY	DESCRIPTION
buttonName	ButtonName	True	Name of the button to unsubscribe.

UnsubscribeButton

Message Type: **response**

PARAMETERS

VALUE	TYPE	MANDATORY	DESCRIPTION
success	Boolean	True	true if successful; false, if failed
resultCode	Result	True	See Result
info	String	False	Provides additional human readable info regarding the result.

SubscribeVehicleData

Message Type: **request**

Subscribes for specific published data items. The data will be only sent if it has changed. The application will be notified by the onVehicleData notification whenever new data is available. To unsubscribe the notifications, use unsubscribe with the same subscriptionType.

PARAMETERS

VALUE	TYPE	MANDATORY	DESCRIPTION
gps	Boolean	False	See GPSData
speed	Boolean	False	The vehicle speed in kilometers per hour
rpm	Boolean	False	The number of revolutions per minute of the engine
fuelLevel	Boolean	False	The fuel level in the tank (percentage). This parameter is deprecated starting RPC Spec 7.0, please see fuelRange.
fuelLevel_State	Boolean	False	The fuel level state. This parameter is deprecated starting RPC Spec 7.0, please see fuelRange.
instantFuelConsumption	Boolean	False	The instantaneous fuel consumption in microlitres

VALUE	TYPE	MANDATORY	DESCRIPTION
fuelRange	Boolean	False	The fuel type, estimated range in KM, fuel level/capacity and fuel level state for the vehicle. See struct FuelRange for details.
climateData	Boolean	False	See ClimateData
externalTemperature	Boolean	False	The external temperature in degrees celsius. This parameter is deprecated starting RPC Spec 7.1, please see climateData.
turnSignal	Boolean	False	See TurnSignal
gearStatus	Boolean	False	See GearStatus
prndl	Boolean	False	See PRNDL. This parameter is deprecated and it is now covered in gearStatus
tirePressure	Boolean	False	See TireStatus
odometer	Boolean	False	Odometer in km

VALUE	TYPE	MANDATORY	DESCRIPTION
beltStatus	Boolean	False	The status of the seat belts
bodyInformation	Boolean	False	The body information including power modes
deviceStatus	Boolean	False	The device status including signal and battery strength
driverBraking	Boolean	False	The status of the brake pedal
wiperStatus	Boolean	False	The status of the wipers
headLampStatus	Boolean	False	Status of the head lamps
engineTorque	Boolean	False	Torque value for engine (in Nm) on non-diesel variants
accPedalPosition	Boolean	False	Accelerator pedal position (percentage depressed)
steeringWheelAngle	Boolean	False	Current angle of the steering wheel (in deg)

VALUE	TYPE	MANDATORY	DESCRIPTION
engineOilLife	Boolean	False	The estimated percentage of remaining oil life of the engine.
electronicParkBrakeStatus	Boolean	False	The status of the park brake as provided by Electric Park Brake (EPB) system.
cloudAppVehicleID	Boolean	False	Parameter used by cloud apps to identify a head unit
stabilityControlsStatus	Boolean	False	See StabilityControlsStatus
eCallInfo	Boolean	False	Emergency Call notification and confirmation data
airbagStatus	Boolean	False	The status of the air bags
emergencyEvent	Boolean	False	Information related to an emergency event (and if it occurred)
clusterModeStatus	Boolean	False	The status modes of the cluster

VALUE	TYPE	MANDATORY	DESCRIPTION
myKey	Boolean	False	Information related to the MyKey feature
windowStatus	Boolean	False	See WindowStatus
handsOffSteering	Boolean	False	To indicate whether driver hands are off the steering wheel
seatOccupancy	Boolean	False	See SeatOccupancy

SubscribeVehicleData

Message Type: **response**

PARAMETERS

VALUE	TYPE	MANDATORY	DESCRIPTION
success	Boolean	True	true, if successful; false, if failed
resultCode	Result	True	See Result
info	String	False	Provides additional human readable info regarding the result.
gps	VehicleDataResult	False	See GPSData
speed	VehicleDataResult	False	The vehicle speed in kilometers per hour
rpm	VehicleDataResult	False	The number of revolutions per minute of the engine
fuelLevel	VehicleDataResult	False	The fuel level in the tank (percentage). This parameter is deprecated starting RPC Spec 7.0, please see fuelRange.
fuelLevel_State	VehicleDataResult	False	The fuel level state. This parameter is deprecated starting RPC Spec 7.0, please see fuelRange.

VALUE	TYPE	MANDATORY	DESCRIPTION
instantFuelConsumption	VehicleDataResult	False	The instantaneous fuel consumption in microlitres
fuelRange	VehicleDataResult	False	The fuel type, estimated range in KM, fuel level/capacity and fuel level state for the vehicle. See struct FuelRange for details.
climateData	VehicleDataResult	False	See ClimateData
externalTemperature	VehicleDataResult	False	The external temperature in degrees celsius. This parameter is deprecated starting RPC Spec 7.1, please see climateData.
turnSignal	VehicleDataResult	False	See TurnSignal
gearStatus	VehicleDataResult	False	See GearStatus
prndl	VehicleDataResult	False	See PRNDL. This parameter is deprecated and it is now covered in gearStatus
tirePressure	VehicleDataResult	False	See TireStatus

VALUE	TYPE	MANDATORY	DESCRIPTION
odometer	VehicleDataResult	False	Odometer in km
beltStatus	VehicleDataResult	False	The status of the seat belts
bodyInformation	VehicleDataResult	False	The body information including power modes
deviceStatus	VehicleDataResult	False	The device status including signal and battery strength
driverBraking	VehicleDataResult	False	The status of the brake pedal
wiperStatus	VehicleDataResult	False	The status of the wipers
headLampStatus	VehicleDataResult	False	Status of the head lamps
engineTorque	VehicleDataResult	False	Torque value for engine (in Nm) on non-diesel variants
accPedalPosition	VehicleDataResult	False	Accelerator pedal position (percentage depressed)

VALUE	TYPE	MANDATORY	DESCRIPTION
steeringWheelAngle	VehicleDataResult	False	Current angle of the steering wheel (in deg)
engineOilLife	VehicleDataResult	False	The estimated percentage of remaining oil life of the engine.
electronicParkBrakeStatus	VehicleDataResult	False	The status of the park brake as provided by Electric Park Brake (EPB) system.
cloudAppVehicleID	VehicleDataResult	False	Parameter used by cloud apps to identify a head unit
stabilityControlsStatus	VehicleDataResult	False	See StabilityControlsStatus
eCallInfo	VehicleDataResult	False	Emergency Call notification and confirmation data
airbagStatus	VehicleDataResult	False	The status of the air bags
emergencyEvent	VehicleDataResult	False	Information related to an emergency event (and if it occurred)

VALUE	TYPE	MANDATORY	DESCRIPTION
clusterModes	VehicleDataResult	False	The status modes of the cluster
myKey	VehicleDataResult	False	Information related to the MyKey feature
windowStatus	VehicleDataResult	False	See WindowStatus
handsOffSteering	VehicleDataResult	False	To indicate whether driver hands are off the steering wheel
seatOccupancy	VehicleDataResult	False	See SeatOccupancy

UnsubscribeVehicleData

Message Type: **request**

This function is used to unsubscribe the notifications from the subscribeVehicleData function.

PARAMETERS

VALUE	TYPE	MANDATORY	DESCRIPTION
gps	Boolean	False	See GPSData
speed	Boolean	False	The vehicle speed in kilometers per hour
rpm	Boolean	False	The number of revolutions per minute of the engine
fuelLevel	Boolean	False	The fuel level in the tank (percentage). This parameter is deprecated starting RPC Spec 7.0, please see fuelRange.
fuelLevel_State	Boolean	False	The fuel level state. This parameter is deprecated starting RPC Spec 7.0, please see fuelRange.
instantFuelConsumption	Boolean	False	The instantaneous fuel consumption in microlitres

VALUE	TYPE	MANDATORY	DESCRIPTION
fuelRange	Boolean	False	The fuel type, estimated range in KM, fuel level/capacity and fuel level state for the vehicle. See struct FuelRange for details.
climateData	Boolean	False	See ClimateData
externalTemperature	Boolean	False	The external temperature in degrees celsius. This parameter is deprecated starting RPC Spec 7.1, please see climateData.
turnSignal	Boolean	False	See TurnSignal
gearStatus	Boolean	False	See GearStatus
prndl	Boolean	False	See PRNDL. This parameter is deprecated and it is now covered in gearStatus
tirePressure	Boolean	False	See TireStatus
odometer	Boolean	False	Odometer in km

VALUE	TYPE	MANDATORY	DESCRIPTION
beltStatus	Boolean	False	The status of the seat belts
bodyInformation	Boolean	False	The body information including power modes
deviceStatus	Boolean	False	The device status including signal and battery strength
driverBraking	Boolean	False	The status of the brake pedal
wiperStatus	Boolean	False	The status of the wipers
headLampStatus	Boolean	False	Status of the head lamps
engineTorque	Boolean	False	Torque value for engine (in Nm) on non-diesel variants
accPedalPosition	Boolean	False	Accelerator pedal position (percentage depressed)
steeringWheelAngle	Boolean	False	Current angle of the steering wheel (in deg)

VALUE	TYPE	MANDATORY	DESCRIPTION
engineOilLife	Boolean	False	The estimated percentage of remaining oil life of the engine.
electronicPark BrakeStatus	Boolean	False	The status of the park brake as provided by Electric Park Brake (EPB) system.
cloudAppVehic leID	Boolean	False	Parameter used by cloud apps to identify a head unit
stabilityControl sStatus	Boolean	False	See StabilityControlsSt atus
eCallInfo	Boolean	False	Emergency Call notification and confirmation data
airbagStatus	Boolean	False	The status of the air bags
emergencyEve nt	Boolean	False	Information related to an emergency event (and if it occurred)
clusterModeSt atus	Boolean	False	The status modes of the cluster

VALUE	TYPE	MANDATORY	DESCRIPTION
myKey	Boolean	False	Information related to the MyKey feature
windowStatus	Boolean	False	See WindowStatus
handsOffSteering	Boolean	False	To indicate whether driver hands are off the steering wheel
seatOccupancy	Boolean	False	See SeatOccupancy

UnsubscribeVehicleData

Message Type: **response**

PARAMETERS

VALUE	TYPE	MANDATORY	DESCRIPTION
success	Boolean	True	true, if successful; false, if failed
resultCode	Result	True	See Result
info	String	False	Provides additional human readable info regarding the result.
gps	VehicleDataResult	False	See GPSData
speed	VehicleDataResult	False	The vehicle speed in kilometers per hour
rpm	VehicleDataResult	False	The number of revolutions per minute of the engine
fuelLevel	VehicleDataResult	False	The fuel level in the tank (percentage). This parameter is deprecated starting RPC Spec 7.0, please see fuelRange.
fuelLevel_State	VehicleDataResult	False	The fuel level state. This parameter is deprecated starting RPC Spec 7.0, please see fuelRange.

VALUE	TYPE	MANDATORY	DESCRIPTION
instantFuelConsumption	VehicleDataResult	False	The instantaneous fuel consumption in microlitres
fuelRange	VehicleDataResult	False	The fuel type, estimated range in KM, fuel level/capacity and fuel level state for the vehicle. See struct FuelRange for details.
climateData	VehicleDataResult	False	See ClimateData
externalTemperature	VehicleDataResult	False	The external temperature in degrees celsius. This parameter is deprecated starting RPC Spec 7.1, please see climateData.
turnSignal	VehicleDataResult	False	See TurnSignal
gearStatus	VehicleDataResult	False	See GearStatus
prndl	VehicleDataResult	False	See PRNDL. This parameter is deprecated and it is now covered in gearStatus
tirePressure	VehicleDataResult	False	See TireStatus

VALUE	TYPE	MANDATORY	DESCRIPTION
odometer	VehicleDataResult	False	Odometer in km
beltStatus	VehicleDataResult	False	The status of the seat belts
bodyInformation	VehicleDataResult	False	The body information including power modes
deviceStatus	VehicleDataResult	False	The device status including signal and battery strength
driverBraking	VehicleDataResult	False	The status of the brake pedal
wiperStatus	VehicleDataResult	False	The status of the wipers
headLampStatus	VehicleDataResult	False	Status of the head lamps
engineTorque	VehicleDataResult	False	Torque value for engine (in Nm) on non-diesel variants
accPedalPosition	VehicleDataResult	False	Accelerator pedal position (percentage depressed)

VALUE	TYPE	MANDATORY	DESCRIPTION
steeringWheelAngle	VehicleDataResult	False	Current angle of the steering wheel (in deg)
engineOilLife	VehicleDataResult	False	The estimated percentage of remaining oil life of the engine.
electronicParkBrakeStatus	VehicleDataResult	False	The status of the park brake as provided by Electric Park Brake (EPB) system.
cloudAppVehicleID	VehicleDataResult	False	Parameter used by cloud apps to identify a head unit
stabilityControlsStatus	VehicleDataResult	False	See StabilityControlsStatus
eCallInfo	VehicleDataResult	False	Emergency Call notification and confirmation data
airbagStatus	VehicleDataResult	False	The status of the air bags
emergencyEvent	VehicleDataResult	False	Information related to an emergency event (and if it occurred)

VALUE	TYPE	MANDATORY	DESCRIPTION
clusterModes	VehicleDataResult	False	The status modes of the cluster
myKey	VehicleDataResult	False	Information related to the MyKey feature
windowStatus	VehicleDataResult	False	See WindowStatus
handsOffSteering	VehicleDataResult	False	To indicate whether driver hands are off the steering wheel
seatOccupancy	VehicleDataResult	False	See SeatOccupancy

GetVehicleData

Message Type: **request**

Non periodic vehicle data read request.

PARAMETERS

VALUE	TYPE	MANDATORY	DESCRIPTION
gps	Boolean	False	See GPSData
speed	Boolean	False	The vehicle speed in kilometers per hour
rpm	Boolean	False	The number of revolutions per minute of the engine
fuelLevel	Boolean	False	The fuel level in the tank (percentage). This parameter is deprecated starting RPC Spec 7.0, please see fuelRange.
fuelLevel_State	Boolean	False	The fuel level state. This parameter is deprecated starting RPC Spec 7.0, please see fuelRange.
instantFuelConsumption	Boolean	False	The instantaneous fuel consumption in microlitres

VALUE	TYPE	MANDATORY	DESCRIPTION
fuelRange	Boolean	False	The fuel type, estimated range in KM, fuel level/capacity and fuel level state for the vehicle. See struct FuelRange for details.
climateData	Boolean	False	See ClimateData
externalTemperature	Boolean	False	The external temperature in degrees celsius. This parameter is deprecated starting RPC Spec 7.1, please see climateData.
turnSignal	Boolean	False	See TurnSignal
vin	Boolean	False	Vehicle identification number
gearStatus	Boolean	False	See GearStatus
prndl	Boolean	False	See PRNDL. This parameter is deprecated and it is now covered in gearStatus
tirePressure	Boolean	False	See TireStatus

VALUE	TYPE	MANDATORY	DESCRIPTION
odometer	Boolean	False	Odometer in km
beltStatus	Boolean	False	The status of the seat belts
bodyInformation	Boolean	False	The body information including ignition status and internal temp
deviceStatus	Boolean	False	The device status including signal and battery strength
driverBraking	Boolean	False	The status of the brake pedal
wiperStatus	Boolean	False	The status of the wipers
headLampStatus	Boolean	False	Status of the head lamps
engineTorque	Boolean	False	Torque value for engine (in Nm) on non-diesel variants
accPedalPosition	Boolean	False	Accelerator pedal position (percentage depressed)

VALUE	TYPE	MANDATORY	DESCRIPTION
steeringWheelAngle	Boolean	False	Current angle of the steering wheel (in deg)
engineOilLife	Boolean	False	The estimated percentage of remaining oil life of the engine.
electronicParkBrakeStatus	Boolean	False	The status of the park brake as provided by Electric Park Brake (EPB) system.
cloudAppVehicleID	Boolean	False	Parameter used by cloud apps to identify a head unit
stabilityControlsStatus	Boolean	False	See StabilityControlsStatus
eCallInfo	Boolean	False	Emergency Call notification and confirmation data
airbagStatus	Boolean	False	The status of the air bags
emergencyEvent	Boolean	False	Information related to an emergency event (and if it occurred)

VALUE	TYPE	MANDATORY	DESCRIPTION
clusterModeStatus	Boolean	False	The status modes of the cluster
myKey	Boolean	False	Information related to the MyKey feature
windowStatus	Boolean	False	See WindowStatus
handsOffSteering	Boolean	False	To indicate whether driver hands are off the steering wheel
seatOccupancy	Boolean	False	See SeatOccupancy

GetVehicleData

Message Type: **response**

PARAMETERS

VALUE	TYPE	MANDATORY	DESCRIPTION
success	Boolean	True	true, if successful; false, if failed
resultCode	Result	True	See Result
info	String	False	Provides additional human readable info regarding the result.
gps	GPSData	False	See GPSData
speed	Float	False	The vehicle speed in kilometers per hour
rpm	Integer	False	The number of revolutions per minute of the engine
fuelLevel	Float	False	The fuel level in the tank (percentage). This parameter is deprecated starting RPC Spec 7.0, please see fuelRange.
fuelLevel_State	ComponentVolumeStatus	False	The fuel level state. This parameter is deprecated starting RPC Spec 7.0, please see fuelRange.

VALUE	TYPE	MANDATORY	DESCRIPTION
instantFuelConsumption	Float	False	The instantaneous fuel consumption in microlitres
fuelRange	FuelRange[]	False	The fuel type, estimated range in KM, fuel level/capacity and fuel level state for the vehicle. See struct FuelRange for details.
climateData	ClimateData	False	See ClimateData
externalTemperature	Float	False	The external temperature in degrees celsius. This parameter is deprecated starting RPC Spec 7.1, please see climateData.
turnSignal	TurnSignal	False	See TurnSignal
vin	String	False	Vehicle identification number
gearStatus	GearStatus	False	See GearStatus

VALUE	TYPE	MANDATORY	DESCRIPTION
prndl	PRNDL	False	See PRNDL. This parameter is deprecated and it is now covered in gearStatus
tirePressure	TireStatus	False	See TireStatus
odometer	Integer	False	Odometer in km
beltStatus	BeltStatus	False	The status of the seat belts
bodyInformation	BodyInformation	False	The body information including power modes
deviceStatus	DeviceStatus	False	The device status including signal and battery strength
driverBraking	VehicleDataEventStatus	False	The status of the brake pedal
wiperStatus	WiperStatus	False	The status of the wipers
headLampStatus	HeadLampStatus	False	Status of the head lamps

VALUE	TYPE	MANDATORY	DESCRIPTION
engineTorque	Float	False	Torque value for engine (in Nm) on non-diesel variants
accPedalPosition	Float	False	Accelerator pedal position (percentage depressed)
steeringWheelAngle	Float	False	Current angle of the steering wheel (in deg)
engineOilLife	Float	False	The estimated percentage of remaining oil life of the engine.
electronicParkBrakeStatus	ElectronicParkBrakeStatus	False	The status of the park brake as provided by Electric Park Brake (EPB) system.
cloudAppVehicleID	String	False	Parameter used by cloud apps to identify a head unit
stabilityControlsStatus	StabilityControlsStatus	False	See StabilityControlsStatus
eCallInfo	ECallInfo	False	Emergency Call notification and confirmation data

VALUE	TYPE	MANDATORY	DESCRIPTION
airbagStatus	AirbagStatus	False	The status of the air bags
emergencyEvent	EmergencyEvent	False	Information related to an emergency event (and if it occurred)
clusterModeStatus	ClusterModeStatus	False	The status modes of the cluster
myKey	MyKey	False	Information related to the MyKey feature
windowStatus	WindowStatus[]	False	See WindowStatus
handsOffSteering	Boolean	False	To indicate whether driver hands are off the steering wheel
seatOccupancy	SeatOccupancy	False	See SeatOccupancy

ReadDID

Message Type: **request**

Non periodic vehicle data read request

PARAMETERS

VALUE	TYPE	MANDATORY	DESCRIPTION
ecuName	Integer	True	Name of ECU.
didLocation	Integer[]	True	Get raw data from vehicle data DID location(s)

ReadDID

Message Type: **response**

PARAMETERS

VALUE	TYPE	MANDATORY	DESCRIPTION
success	Boolean	True	true, if successful; false, if failed
resultCode	Result	True	See Result
info	String	False	Provides additional human readable info regarding the result.
didResult	DIDResult[]	False	Array of requested DID results (with data if available).

GetDTCs

Message Type: **request**

Vehicle module diagnostic trouble code request.

PARAMETERS

VALUE	TYPE	MANDATORY	DESCRIPTION
ecuName	Integer	True	Name of ECU.
dtcMask	Integer	False	DTC Mask Byte to be sent in diagnostic request to module .

GetDTCs

Message Type: **response**

PARAMETERS

VALUE	TYPE	MANDATORY	DESCRIPTION
success	Boolean	True	true, if successful; false, if failed
resultCode	Result	True	See Result
info	String	False	Provides additional human readable info regarding the result.
ecuHeader	Integer	False	2 byte ECU Header for DTC response (as defined in VHR_Layout_Specification_DTCs.pdf)
dtc	String[]	False	Array of all reported DTCs on module (ecuHeader contains information if list is truncated). Each DTC is represented by 4 bytes (3 bytes of data and 1 byte status as defined in VHR_Layout_Specification_DTCs.pdf).

DiagnosticMessage

Message Type: **request**

Non periodic vehicle diagnostic request

VALUE	TYPE	MANDATORY	DESCRIPTION
targetID	Integer	True	Name of target ECU.
messageLength	Integer	True	Length of message (in bytes).
messageData	Integer[]	True	Array of bytes comprising CAN message.

DiagnosticMessage

Message Type: **response**

PARAMETERS

VALUE	TYPE	MANDATORY	DESCRIPTION
success	Boolean	True	true, if successful; false, if failed
resultCode	Result	True	See Result
info	String	False	Provides additional human readable info regarding the result.
messageDataResult	Integer[]	False	Array of bytes comprising CAN message result.

ScrollableMessage

Message Type: **request**

Creates a full screen overlay containing a large block of formatted text that can be scrolled with up to 8 SoftButtons defined

PARAMETERS

VALUE	TYPE	MANDATORY	DESCRIPTION
scrollableMessageBody	String	True	Body of text that can include newlines and tabs.
timeout	Integer	False	App defined timeout. Indicates how long of a timeout from the last action (i.e. scrolling message resets timeout).
softButtons	SoftButton[]	False	App defined SoftButtons. If omitted on supported displays, only the system defined "Close" SoftButton will be displayed.
cancelID	Integer	False	An ID for this specific ScrollableMessage to allow cancellation through the CancelInteraction RPC.

ScrollableMessage

Message Type: **response**

PARAMETERS

VALUE	TYPE	MANDATORY	DESCRIPTION
success	Boolean	True	true, if successful; false, if failed
resultCode	Result	True	See Result
info	String	False	Provides additional human readable info regarding the result.

Slider

Message Type: **request**

Creates a full screen or pop-up overlay (depending on platform) with a single user controlled slider.

PARAMETERS

VALUE	TYPE	MANDATORY	DESCRIPTION
numTicks	Integer	True	Number of selectable items on a horizontal axis
position	Integer	True	Initial position of slider control (cannot exceed numTicks)
sliderHeader	String	True	Text header to display
sliderFooter	String[]	False	Text footer to display (meant to display min/max threshold descriptors). For a static text footer, only one footer string shall be provided in the array. For a dynamic text footer, the number of footer text string in the array must match the numTicks value. For a dynamic text footer, text array string should correlate with potential slider position index. If omitted on supported displays, no footer text shall be displayed.

VALUE	TYPE	MANDATORY	DESCRIPTION
timeout	Integer	False	App defined timeout. Indicates how long of a timeout from the last action (i.e. sliding control resets timeout). If omitted, the value is set to 10000.
cancelID	Integer	False	An ID for this specific Slider to allow cancellation through the CancelInteraction RPC.

Slider

Message Type: **response**

PARAMETERS

VALUE	TYPE	MANDATORY	DESCRIPTION
success	Boolean	True	true, if successful; false, if failed
resultCode	Result	True	See Result
info	String	False	Provides additional human readable info regarding the result.
sliderPosition	Integer	False	Current slider value returned when saved or canceled (aborted) This value is only returned for resultCodes "SAVED" or "ABORTED"

ShowConstantTBT

Message Type: **request**

PARAMETERS

VALUE	TYPE	MANDATORY	DESCRIPTION
navigationText 1	String	False	
navigationText 2	String	False	
eta	String	False	
timeToDestination	String	False	
totalDistance	String	False	
turnIcon	Image	False	
nextTurnIcon	Image	False	
distanceToManeuver	Float	False	Distance (in meters) until next maneuver. May be used to calculate progress bar.
distanceToManeuverScale	Float	False	Distance (in meters) from previous maneuver to next maneuver. May be used to calculate progress bar.

VALUE	TYPE	MANDATORY	DESCRIPTION
maneuverComplete	Boolean	False	If and when a maneuver has completed while an AlertManeuver is active, the app must send this value set to TRUE in order to clear the AlertManeuver overlay. If omitted the value will be assumed as FALSE.
softButtons	SoftButton[]	False	Three dynamic SoftButtons available (first SoftButton is fixed to "Turns"). If omitted on supported displays, the currently displayed SoftButton values will not change.

ShowConstantTBT

Message Type: **response**

PARAMETERS

VALUE	TYPE	MANDATORY	DESCRIPTION
success	Boolean	True	true, if successful; false, if failed
resultCode	Result	True	See Result
info	String	False	Provides additional human readable info regarding the result.

AlertManeuver

Message Type: **request**

PARAMETERS

VALUE	TYPE	MANDATORY	DESCRIPTION
ttsChunks	TTSChunk[]	False	An array of text chunks of type TTSChunk. See TTSChunk
softButtons	SoftButton[]	False	If omitted on supported displays, only the system defined "Close" SoftButton shall be displayed.

AlertManeuver

Message Type: **response**

PARAMETERS

VALUE	TYPE	MANDATORY	DESCRIPTION
success	Boolean	True	true, if successful; false, if failed
resultCode	Result	True	See Result
info	String	False	Provides additional human readable info regarding the result.

UpdateTurnList

Message Type: **request**

PARAMETERS

VALUE	TYPE	MANDATORY	DESCRIPTION
turnList	Turn[]	False	
softButtons	SoftButton[]	False	If omitted on supported displays, app-defined SoftButton will be left blank.

UpdateTurnList

Message Type: **response**

PARAMETERS

VALUE	TYPE	MANDATORY	DESCRIPTION
success	Boolean	True	true, if successful; false, if failed
resultCode	Result	True	See Result
info	String	False	Provides additional human readable info regarding the result.

ChangeRegistration

Message Type: **request**

PARAMETERS

VALUE	TYPE	MANDATORY	DESCRIPTION
language	Language	True	Requested voice engine (VR+TTS) language registration
hmiDisplayLanguage	Language	True	Request display language registration
appName	String	False	Request new app name registration
ttsName	TTSChunk[]	False	Request new ttsName registration
ngnMediaScreenAppName	String	False	Request new app short name registration
vrSynonyms	String[]	False	Request new VR synonyms registration

ChangeRegistration

Message Type: **response**

PARAMETERS

VALUE	TYPE	MANDATORY	DESCRIPTION
success	Boolean	True	true, if successful; false, if failed
resultCode	Result	True	See Result
info	String	False	Provides additional human readable info regarding the result.

GenericResponse

Message Type: **response**

Generic Response is sent, when the name of a received msg cannot be retrieved. Only used in case of an error. Currently, only resultCode INVALID_DATA is used.

PARAMETERS

VALUE	TYPE	MANDATORY	DESCRIPTION
success	Boolean	True	true, if successful; false, if failed
resultCode	Result	True	See Result
info	String	False	Provides additional human readable info regarding the result.

PutFile

Message Type: **request**

Used to push a binary data onto the module from a mobile device, such as icons and album art Not supported on first generation of SDL enabled modules. Binary data is in binary part of hybrid msg.

PARAMETERS

VALUE	TYPE	MANDATORY	DESCRIPTION
syncFileName	String	True	File reference name.
fileType	FileType	True	Selected file type.
persistentFile	Boolean	False	Indicates if the file is meant to persist between sessions / ignition cycles. If set to TRUE, then the system will aim to persist this file through session / cycles. While files with this designation will have priority over others, they are subject to deletion by the system at any time. In the event of automatic deletion by the system, the app will receive a rejection and have to resend the file. If omitted, the value will be set to false.

VALUE	TYPE	MANDATORY	DESCRIPTION
systemFile	Boolean	False	Indicates if the file is meant to be passed thru core to elsewhere on the system. If set to TRUE, then the system will instead pass the data thru as it arrives to a predetermined area outside of core. If omitted, the value will be set to false.
offset	Integer	False	Optional offset in bytes for resuming partial data chunks
length	Integer	False	Optional length in bytes for resuming partial data chunks If offset is set to 0, then length is the total length of the file to be downloaded
crc	Integer	False	Additional CRC32 checksum to protect data integrity up to 512 Mbits

PutFile

Message Type: **response**

Response is sent, when the file data was copied (success case). Or when an error occurred. Not supported on first generation SDL enabled vehicles.

PARAMETERS

VALUE	TYPE	MANDATORY	DESCRIPTION
success	Boolean	True	true, if successful; false, if failed
resultCode	Result	True	See Result
spaceAvailable	Integer	False	Provides the total local space available in SDL Core for the registered app. If the transfer has systemFile enabled, then the value will be set to 0 automatically.
info	String	False	Provides additional human readable info regarding the result.

GetFile

Message Type: **request**

This request is sent to the module to retrieve a file

PARAMETERS

VALUE	TYPE	MANDATORY	DESCRIPTION
fileName	String	True	File name that should be retrieved
appServiceId	String	False	ID of the service that should have uploaded the requested file.
fileType	FileType	False	Selected file type.
offset	Integer	False	Optional offset in bytes for resuming partial data chunks
length	Integer	False	Optional length in bytes for resuming partial data chunks If offset is set to 0, then length is the total length of the file to be retrieved

GetFile

Message Type: **response**

This response includes the data that is requested from the specific service

PARAMETERS

VALUE	TYPE	MANDATORY	DESCRIPTION
success	Boolean	True	true, if successful; false, if failed
resultCode	Result	True	See Result
info	String	False	Provides additional human readable info regarding the result.
offset	Integer	False	Optional offset in bytes for resuming partial data chunks
length	Integer	False	Optional length in bytes for resuming partial data chunks if offset is set to 0, then length is the total length of the file to be downloaded
fileType	FileType	False	File type that is being sent in response.
crc	Integer	False	Additional CRC32 checksum to protect data integrity up to 512 Mbits

DeleteFile

Message Type: **request**

Used to delete a file resident on the module in the app's local cache. Not supported on first generation SDL enabled vehicles.

PARAMETERS

VALUE	TYPE	MANDATORY	DESCRIPTION
syncFileName	String	True	File reference name.

DeleteFile

Message Type: **response**

Response is sent, when the file data was deleted (success case). Or when an error occurred. Not supported on First generation SDL enabled vehicles.

PARAMETERS

VALUE	TYPE	MANDATORY	DESCRIPTION
success	Boolean	True	true if successful; false, if failed
resultCode	Result	True	See Result
spaceAvailable	Integer	False	Provides the total local space available on the module for the registered app.
info	String	False	Provides additional human readable info regarding the result.

ListFiles

Message Type: **request**

Requests the current list of resident filenames for the registered app. Not supported on first generation SDL enabled vehicles.

ListFiles

Message Type: **response**

Returns the current list of resident filenames for the registered app along with the current space available Not supported on First generation SDL enabled vehicles.

PARAMETERS

VALUE	TYPE	MANDATORY	DESCRIPTION
success	Boolean	True	true, if successful; false, if failed
resultCode	Result	True	See Result
filenames	String[]	False	An array of all filenames resident on the module for the given registered app. If omitted, then no files currently reside on the system.
spaceAvailable	Integer	False	Provides the total local space available on the module for the registered app.
info	String	False	Provides additional human readable info regarding the result.

SetAppIcon

Message Type: **request**

Used to set existing local file on the module as the app's icon Not supported on first generation SDL enabled vehicles.

PARAMETERS

VALUE	TYPE	MANDATORY	DESCRIPTION
syncFileName	String	True	File reference name.

SetAppIcon

Message Type: **response**

Response is sent, when the file data was copied (success case). Or when an error occurred. Not supported on First generation SDL enabled vehicles.

PARAMETERS

VALUE	TYPE	MANDATORY	DESCRIPTION
success	Boolean	True	true, if successful; false, if failed
resultCode	Result	True	See Result
info	String	False	Provides additional human readable info regarding the result.

SetDisplayLayout

Message Type: **request**
Deprecated since: 6.0.0
This RPC is deprecated. Use Show RPC to change layout.

PARAMETERS

VALUE	TYPE	MANDATORY	DESCRIPTION
displayLayout	String	True	Predefined or dynamically created screen layout. Currently only predefined screen layouts are defined.
dayColorScheme	TemplateColorScheme	False	
nightColorScheme	TemplateColorScheme	False	

SetDisplayLayout

Message Type: **response**
Deprecated since: 6.0.0
This RPC is deprecated. Use Show RPC to change layout.

PARAMETERS

VALUE	TYPE	MANDATORY	DESCRIPTION
success	Boolean	True	true, if successful; false, if failed
resultCode	Result	True	See Result
displayCapabilities	DisplayCapabilities	False	See DisplayCapabilities
buttonCapabilities	ButtonCapabilities[]	False	See ButtonCapabilities
softButtonCapabilities	SoftButtonCapabilities[]	False	If returned, the platform supports on-screen SoftButtons; see SoftButtonCapabilities.
presetBankCapabilities	PresetBankCapabilities	False	If returned, the platform supports custom on-screen Presets; see PresetBankCapabilities.
info	String	False	Provides additional human readable info regarding the result.

SystemRequest

Message Type: **request**

An asynchronous request from the device; binary data can be included in hybrid part of message for some requests (such as HTTP, Proprietary, or Authentication requests)

PARAMETERS

VALUE	TYPE	MANDATORY	DESCRIPTION
requestType	RequestType	True	The type of system request. Note that Proprietary requests should forward the binary data to the known proprietary module on the system.
requestSubType	String	False	This parameter is filled for supporting OEM proprietary data exchanges.
fileName	String	False	Filename of HTTP data to store in predefined system staging area. Mandatory if requestType is HTTP. PROPRIETARY requestType should ignore this parameter.

SystemRequest

Message Type: **response**

PARAMETERS

VALUE	TYPE	MANDATORY	DESCRIPTION
success	Boolean	True	true, if successful; false, if failed
resultCode	Result	True	See Result
info	String	False	Provides additional human readable info regarding the result.

SendLocation

Message Type: **request**

PARAMETERS

VALUE	TYPE	MANDATORY	DESCRIPTION
longitudeDegrees	Float	False	
latitudeDegrees	Float	False	
locationName	String	False	Name / title of intended location
locationDescription	String	False	Description intended location / establishment (if applicable)
addressLines	String[]	False	Location address (if applicable)
phoneNumber	String	False	Phone number of intended location / establishment (if applicable)
locationImage	Image	False	Image / icon of intended location (if applicable and supported)
timeStamp	DateTime	False	timestamp in ISO 8601 format
address	OASISAddress	False	Address to be used for setting destination
VALUE	TYPE	MANDATORY	DESCRIPTION

VALUE	TYPE	MANDATORY	DESCRIPTION
deliveryMode	DeliveryMode	False	Defines the mode of prompt for user

SendLocation

Message Type: **response**

PARAMETERS

VALUE	TYPE	MANDATORY	DESCRIPTION
success	Boolean	True	true, if successful; false, if failed
resultCode	Result	True	See Result
info	String	False	Provides additional human readable info regarding the result.

DialNumber

Message Type: **request**

Dials a phone number and switches to phone application.

PARAMETERS

VALUE	TYPE	MANDATORY	DESCRIPTION
number	String	True	Phone number is a string, which can be up to 40 chars. All characters shall be stripped from string except digits 0-9 and * # , ; +

DialNumber

Message Type: **response**

PARAMETERS

VALUE	TYPE	MANDATORY	DESCRIPTION
success	Boolean	True	true, if successful; false, if failed
resultCode	Result	True	See Result
info	String	False	Provides additional human readable info regarding the result.

ButtonPress

Message Type: **request**

NOTE: Certain ButtonNames are tied to specific RC module types. See ButtonName

PARAMETERS

VALUE	TYPE	MANDATORY	DESCRIPTION
moduleType	ModuleType	True	The module where the button should be pressed
moduleId	String	False	Id of a module in the published ButtonCapabilities
buttonName	ButtonName	True	The name of the supported RC button.
buttonPressMode	ButtonPressMode	True	Indicates whether this is a LONG or SHORT button press event.

ButtonPress

Message Type: **response**

PARAMETERS

VALUE	TYPE	MANDATORY	DESCRIPTION
resultCode	Result	True	See Result
info	String	False	
success	Boolean	True	true if successful; false, if failed

GetInteriorVehicleData

Message Type: **request**

PARAMETERS

VALUE	TYPE	MANDATORY	DESCRIPTION
moduleType	ModuleType	True	The type of an RC module to retrieve module data from the vehicle. In the future, this should be the Identification of a module.
moduleId	String	False	Id of a module, published by System Capability.
subscribe	Boolean	False	If subscribe is true, the head unit will register OnInteriorVehicleData notifications for the requested module (moduleId and moduleType). If subscribe is false, the head unit will unregister OnInteriorVehicleData notifications for the requested module (moduleId and moduleType). If subscribe is not included, the subscription status of the app for the requested module (moduleId and moduleType) will remain unchanged.

GetInteriorVehicleData

Message Type: **response**

PARAMETERS

VALUE	TYPE	MANDATORY	DESCRIPTION
moduleData	ModuleData	False	
resultCode	Result	True	See Result
info	String	False	
success	Boolean	True	true if successful; false, if failed
isSubscribed	Boolean	False	It is a conditional-mandatory parameter: must be returned in case "subscribe" parameter was present in the related request. if "true" - the "moduleType" from request is successfully subscribed and the head unit will send onInteriorVehicleData notifications for the moduleType. if "false" - the "moduleType" from request is either unsubscribed or failed to subscribe.

GetInteriorVehicleDataConsent

Message Type: **request**

PARAMETERS

VALUE	TYPE	MANDATORY	DESCRIPTION
moduleType	ModuleType	True	The module type that the app requests to control.
moduleIds	String[]	True	Ids of a module of same type, published by System Capability.

GetInteriorVehicleDataConsent

Message Type: **response**

PARAMETERS

VALUE	TYPE	MANDATORY	DESCRIPTION
success	Boolean	True	true if successful; false, if failed
resultCode	Result	True	See Result
info	String	False	
allowed	Boolean[]	False	This array has the same size as "moduleIds" in the request and each element corresponds to one moduleId If true, SDL grants the permission for the requested module If false, SDL denies the permission for the requested module.

ReleaseInteriorVehicleDataModule

Message Type: **request**

PARAMETERS

VALUE	TYPE	MANDATORY	DESCRIPTION
moduleType	ModuleType	True	
moduleId	String	False	Id of a module, published by System Capability.

ReleaseInteriorVehicleDataModule

Message Type: **response**

PARAMETERS

VALUE	TYPE	MANDATORY	DESCRIPTION
success	Boolean	True	true if successful; false, if failed
resultCode	Result	True	See Result
info	String	False	

SetInteriorVehicleData

Message Type: **request**

PARAMETERS

VALUE	TYPE	MANDATORY	DESCRIPTION
moduleData	ModuleData	True	The module data to set for the requested RC module.

SetInteriorVehicleData

Message Type: **response**

Used to set the values of one remote control module

PARAMETERS

VALUE	TYPE	MANDATORY	DESCRIPTION
moduleData	ModuleData	False	
resultCode	Result	True	See Result
info	String	False	
success	Boolean	True	true if successful; false, if failed

SubscribeWayPoints

Message Type: **request**

To subscribe in getting changes for Waypoints/destinations

SubscribeWayPoints

Message Type: **response**

PARAMETERS

VALUE	TYPE	MANDATORY	DESCRIPTION
success	Boolean	True	true, if successful; false, if failed
resultCode	Result	True	See Result
info	String	False	Provides additional human readable info regarding the result.

GetWayPoints

Message Type: **request**

Request for getting waypoint/destination data.

PARAMETERS

VALUE	TYPE	MANDATORY	DESCRIPTION
wayPointType	WayPointType	True	To request for either the destination only or for all waypoints including destination

GetWayPoints

Message Type: **response**

PARAMETERS

VALUE	TYPE	MANDATORY	DESCRIPTION
success	Boolean	True	true, if successful; false, if failed
resultCode	Result	True	See Result
info	String	False	Provides additional human readable info regarding the result.
wayPoints	LocationDetails[]	False	See LocationDetails

UnsubscribeWayPoints

Message Type: **request**

Request to unsubscribe from WayPoints and Destination

UnsubscribeWayPoints

Message Type: **response**

PARAMETERS

VALUE	TYPE	MANDATORY	DESCRIPTION
success	Boolean	True	true, if successful; false, if failed
resultCode	Result	True	See Result
info	String	False	Provides additional human readable info regarding the result.
wayPoints	LocationDetails[]	False	See LocationDetails

GetSystemCapability

Message Type: **request**

Request for expanded information about a supported system/HMI capability

PARAMETERS

VALUE	TYPE	MANDATORY	DESCRIPTION
systemCapabilityType	SystemCapabilityType	True	The type of system capability to get more information on
subscribe	Boolean	False	Flag to subscribe to updates of the supplied service capability type. If true, the requester will be subscribed. If false, the requester will not be subscribed and be removed as a subscriber if it was previously subscribed.

GetSystemCapability

Message Type: **response**

PARAMETERS

VALUE	TYPE	MANDATORY	DESCRIPTION
systemCapability	SystemCapability	False	
resultCode	Result	True	See Result
info	String	False	Provides additional human readable info regarding the result.
success	Boolean	True	true if successful; false, if failed

SendHapticData

Message Type: **request**

Send the spatial data gathered from SDLCarWindow or VirtualDisplayEncoder to the HMI. This data will be utilized by the HMI to determine how and when haptic events should occur

PARAMETERS

VALUE	TYPE	MANDATORY	DESCRIPTION
hapticRectData	HapticRect[]	False	Array of spatial data structures that represent the locations of all user controls present on the HMI. This data should be updated if/when the application presents a new screen. When a request is sent, if successful, it will replace all spatial data previously sent through RPC. If an empty array is sent, the existing spatial data will be cleared

SendHapticData

Message Type: **response**

PARAMETERS

VALUE	TYPE	MANDATORY	DESCRIPTION
success	Boolean	True	true if successful; false if failed
info	String	False	Provides additional human readable info regarding the result.
resultCode	Result	True	See Result

SetCloudAppProperties

Message Type: **request**

RPC used to enable/disable a cloud application and set its cloud-related policy properties

PARAMETERS

VALUE	TYPE	MANDATORY	DESCRIPTION
properties	CloudAppProperties	True	The new cloud application properties

SetCloudAppProperties

Message Type: **response**

The response to SetCloudAppProperties

PARAMETERS

VALUE	TYPE	MANDATORY	DESCRIPTION
success	Boolean	True	true if successful; false if failed
resultCode	Result	True	See Result
info	String	False	Provides additional human readable info regarding the result.

GetCloudAppProperties

Message Type: **request**

RPC used to get the current properties of a cloud application

PARAMETERS

VALUE	TYPE	MANDATORY	DESCRIPTION
appID	String	True	

GetCloudAppProperties

Message Type: **response**

The response to GetCloudAppProperties

PARAMETERS

VALUE	TYPE	MANDATORY	DESCRIPTION
properties	CloudAppProperties	False	The requested cloud application properties
success	Boolean	True	true if successful; false if failed
resultCode	Result	True	See Result
info	String	False	Provides additional human readable info regarding the result.

PublishAppService

Message Type: **request**

Registers a service offered by this app on the module. Subsequent calls with the same service type will update the manifest for that service.

PARAMETERS

VALUE	TYPE	MANDATORY	DESCRIPTION
appServiceManifest	AppServiceManifest	True	The manifest of the service that wishes to be published. If already published, the updated manifest for this service.

PublishAppService

Message Type: **response**

Response to the request to register a service offered by this app on the module

PARAMETERS

VALUE	TYPE	MANDATORY	DESCRIPTION
success	Boolean	True	true, if successful; false, if failed
resultCode	Result	True	See Result
info	String	False	Provides additional human readable info regarding the result.
appServiceRecord	AppServiceRecord	False	If the request was successful, this object will be the current status of the service record for the published service. This will include the Core supplied service ID.

UnpublishAppService

Message Type: **request**

Unpublish an existing service published by this application.

PARAMETERS

VALUE	TYPE	MANDATORY	DESCRIPTION
serviceID	String	True	The ID of the service to be unpublished.

UnpublishAppService

Message Type: **response**

The response to UnpublishAppService

PARAMETERS

VALUE	TYPE	MANDATORY	DESCRIPTION
success	Boolean	True	true, if successful; false, if failed
resultCode	Result	True	See Result
info	String	False	Provides additional human readable info regarding the result.

GetAppServiceData

Message Type: **request**

This request asks the module for current data related to the specific service. It also includes an option to subscribe to that service for future updates

PARAMETERS

VALUE	TYPE	MANDATORY	DESCRIPTION
serviceType	String	True	The type of service that is to be offered by this app. See AppServiceType for known enum equivalent types. Parameter is a string to allow for new service types to be used by apps on older versions of SDL Core.
subscribe	Boolean	False	If true, the consumer is requesting to subscribe to all future updates from the service publisher. If false, the consumer doesn't wish to subscribe and should be unsubscribed if it was previously subscribed.

GetAppServiceData

Message Type: **response**

This response includes the data that was requested from the specific service

PARAMETERS

VALUE	TYPE	MANDATORY	DESCRIPTION
success	Boolean	True	true, if successful; false, if failed
resultCode	Result	True	See Result
info	String	False	Provides additional human readable info regarding the result.
serviceData	AppServiceData	False	

PerformAppServiceInteraction

Message Type: **request**

PARAMETERS

VALUE	TYPE	MANDATORY	DESCRIPTION
serviceUri	String	True	Fully qualified URI based on a predetermined scheme provided by the app service. SDL makes no guarantee that this URI is correct.
serviceID	String	True	The service ID that the app consumer wishes to send this URI.
originApp	String	True	This string is the appID of the app requesting the app service provider take the specific action.
requestService Active	Boolean	False	This flag signals the requesting consumer would like this service to become the active primary service of the destination's type.

PerformAppServiceInteraction

Message Type: **response**

PARAMETERS

VALUE	TYPE	MANDATORY	DESCRIPTION
success	Boolean	True	true, if successful; false, if failed
resultCode	Result	True	See Result. All results will be available for this response.
info	String	False	Provides additional human readable info regarding the result.
serviceSpecific Result	String	False	The service can provide specific result strings to the consumer through this param.

CancelInteraction

Message Type: **request**

Close an active interaction on the HMI.

PARAMETERS

VALUE	TYPE	MANDATORY	DESCRIPTION
cancelID	Integer	False	The ID of the specific interaction you want to dismiss. If not set, the most recent of the RPC type set in functionID will be dismissed.
functionID	Integer	True	The ID of the type of interaction the developer wants to dismiss. Only values 10, (PerformInteraction ID), 12 (AlertID), 25 (ScrollableMessage ID), 26 (SliderID), and 64 (SubtleAlertID) are permitted.

CancelInteraction

Message Type: **response**

If no applicable request can be dismissed, the result will be IGNORED.

PARAMETERS

VALUE	TYPE	MANDATORY	DESCRIPTION
success	Boolean	True	true if successful; false, if failed
resultCode	Result	True	See Result
info	String	False	Provides additional human readable info regarding the result.

CloseApplication

Message Type: **request**

Request from the application to exit the foreground and enter HMI_NONE.

CloseApplication

Message Type: **response**

PARAMETERS

VALUE	TYPE	MANDATORY	DESCRIPTION
success	Boolean	True	true if successful; false, if failed
resultCode	Result	True	See Result
info	String	False	Provides additional human readable info regarding the result.

OnHMIStatus

Message Type: **notification**

PARAMETERS

VALUE	TYPE	MANDATORY	DESCRIPTION
hmiLevel	HMILevel	True	See HMILevel
audioStreamingState	AudioStreamingState	True	See AudioStreamingState
systemContext	SystemContext	True	See SystemContext
videoStreamingState	VideoStreamingState	False	See VideoStreamingState. If it is NOT_STREAMABLE, the app must stop streaming video to SDL Core(stop service).
windowID	Integer	False	This is the unique ID assigned to the window that this RPC is intended. If this param is not included, it will be assumed that this request is specifically for the main window on the main display. See PredefinedWindows enum.

OnAppInterfaceUnregistered

Message Type: **notification**

PARAMETERS

VALUE	TYPE	MANDATORY	DESCRIPTION
reason	AppInterfaceUnregisteredReason	True	See AppInterfaceUnregisteredReason

OnButtonEvent

Message Type: **notification**

Notifies application of UP/DOWN events for buttons to which the application is subscribed.

PARAMETERS

VALUE	TYPE	MANDATORY	DESCRIPTION
buttonName	ButtonName	True	
buttonEventMode	ButtonEventMode	True	Indicates whether this is an UP or DOWN event.
customButtonID	Integer	False	If ButtonName is "CUSTOM_BUTTON", this references the integer ID passed by a custom button. (e.g. softButton ID)

OnButtonPress

Message Type: **notification**

Notifies application of LONG/SHORT press events for buttons to which the application is subscribed.

PARAMETERS

VALUE	TYPE	MANDATORY	DESCRIPTION
buttonName	ButtonName	True	
buttonPressMode	ButtonPressMode	True	Indicates whether this is a LONG or SHORT button press event.
customButtonID	Integer	False	If ButtonName is "CUSTOM_BUTTON", this references the integer ID passed by a custom button. (e.g. softButton ID)

OnVehicleData

Message Type: **notification**

Callback for the periodic and non periodic vehicle data read function.

PARAMETERS

VALUE	TYPE	MANDATORY	DESCRIPTION
gps	GPSData	False	See GPSData
speed	Float	False	The vehicle speed in kilometers per hour
rpm	Integer	False	The number of revolutions per minute of the engine
fuelLevel	Float	False	The fuel level in the tank (percentage). This parameter is deprecated starting RPC Spec 7.0, please see fuelRange.
fuelLevel_State	ComponentVolumeStatus	False	The fuel level state. This parameter is deprecated starting RPC Spec 7.0, please see fuelRange.
instantFuelConsumption	Float	False	The instantaneous fuel consumption in microlitres

VALUE	TYPE	MANDATORY	DESCRIPTION
fuelRange	FuelRange[]	False	The fuel type, estimated range in KM, fuel level/capacity and fuel level state for the vehicle. See struct FuelRange for details.
climateData	ClimateData	False	See ClimateData
externalTemperature	Float	False	The external temperature in degrees celsius. This parameter is deprecated starting RPC Spec 7.1, please see climateData.
turnSignal	TurnSignal	False	See TurnSignal
vin	String	False	Vehicle identification number.
gearStatus	GearStatus	False	See GearStatus
prndl	PRNDL	False	See PRNDL. This parameter is deprecated and it is now covered in gearStatus
tirePressure	TireStatus	False	See TireStatus

VALUE	TYPE	MANDATORY	DESCRIPTION
odometer	Integer	False	Odometer in km
beltStatus	BeltStatus	False	The status of the seat belts
bodyInformation	BodyInformation	False	The body information including power modes
deviceStatus	DeviceStatus	False	The device status including signal and battery strength
driverBraking	VehicleDataEventStatus	False	The status of the brake pedal
wiperStatus	WiperStatus	False	The status of the wipers
headLampStatus	HeadLampStatus	False	Status of the head lamps
engineTorque	Float	False	Torque value for engine (in Nm) on non-diesel variants
accPedalPosition	Float	False	Accelerator pedal position (percentage depressed)

VALUE	TYPE	MANDATORY	DESCRIPTION
steeringWheelAngle	Float	False	Current angle of the steering wheel (in deg)
engineOilLife	Float	False	The estimated percentage of remaining oil life of the engine.
electronicParkBrakeStatus	ElectronicParkBrakeStatus	False	The status of the park brake as provided by Electric Park Brake (EPB) system.
cloudAppVehicleID	String	False	Parameter used by cloud apps to identify a head unit
stabilityControlsStatus	StabilityControlsStatus	False	See StabilityControlsStatus
eCallInfo	ECallInfo	False	Emergency Call notification and confirmation data
airbagStatus	AirbagStatus	False	The status of the air bags
emergencyEvent	EmergencyEvent	False	Information related to an emergency event (and if it occurred)

VALUE	TYPE	MANDATORY	DESCRIPTION

VALUE	TYPE	MANDATORY	DESCRIPTION
clusterModeStatus	ClusterModeStatus	False	The status modes of the cluster
myKey	MyKey	False	Information related to the MyKey feature
windowStatus	WindowStatus[]	False	See WindowStatus
handsOffSteering	Boolean	False	To indicate whether driver hands are off the steering wheel
seatOccupancy	SeatOccupancy	False	See SeatOccupancy

OnCommand

Message Type: **notification**

PARAMETERS

VALUE	TYPE	MANDATORY	DESCRIPTION
cmdID	Integer	True	Command ID, which is related to a specific menu entry
triggerSource	TriggerSource	True	See TriggerSource

OnTBTClientState

Message Type: **notification**

Provides applications with notifications specific to the current TBT client status on the module

PARAMETERS

VALUE	TYPE	MANDATORY	DESCRIPTION
state	TBTState	True	Current State of TBT client

OnDriverDistraction

Message Type: **notification**

Provides driver distraction state to mobile applications

PARAMETERS

VALUE	TYPE	MANDATORY	DESCRIPTION
state	DriverDistractionState	True	Current State of Driver Distraction
lockScreenDismissalEnabled	Boolean	False	If enabled, the lock screen will be able to be dismissed while connected to SDL, allowing users the ability to interact with the app. Dismissals should include a warning to the user and ensure that they are not the driver.
lockScreenDismissalWarning	String	False	Warning message to be displayed on the lock screen when dismissal is enabled. This warning should be used to ensure that the user is not the driver of the vehicle, ex. Swipe down to dismiss, acknowledging that you are not the driver.. This parameter must be present if "lockScreenDismissalEnabled" is set to true.

OnPermissionsChange

Message Type: **notification**

Provides update to app of which policy-table-enabled functions are available

PARAMETERS

VALUE	TYPE	MANDATORY	DESCRIPTION
permissionItem	PermissionItem[]	True	Change in permissions for a given set of RPCs
requireEncryption	Boolean	False	

OnAudioPassThru

Message Type: **notification**

Binary data is in binary part of hybrid msg

OnLanguageChange

Message Type: **notification**

PARAMETERS

VALUE	TYPE	MANDATORY	DESCRIPTION
language	Language	True	Current SDL voice engine (VR+TTS) language
hmiDisplayLanguage	Language	True	Current display language

OnKeyboardInput

Message Type: **notification**

On-screen keyboard event. Can be full string or individual keypresses depending on keyboard mode.

PARAMETERS

VALUE	TYPE	MANDATORY	DESCRIPTION
event	KeyboardEvent	True	On-screen keyboard input data.
data	String	False	On-screen keyboard input data. For dynamic keypress events, this will be the current compounded string of entry text. For entry submission events, this will be the full text entry (this will always return regardless of the mode). For entry cancelled and entry aborted events, this data param will be omitted.

OnTouchEvent

Message Type: **notification**

Notifies about touch events on the screen's prescribed area

PARAMETERS

VALUE	TYPE	MANDATORY	DESCRIPTION
type	TouchType	True	The type of touch event.
event	TouchEvent[]	True	List of all individual touches involved in this event.

OnSystemRequest

Message Type: **notification**

An asynchronous request from the system for specific data from the device or the cloud or response to a request from the device or cloud Binary data can be included in hybrid part of message for some requests (such as Authentication request responses)

PARAMETERS

VALUE	TYPE	MANDATORY	DESCRIPTION
requestType	RequestType	True	The type of system request.
requestSubType	String	False	This parameter is filled for supporting OEM proprietary data exchanges.
url	String	False	Optional URL for HTTP requests. If blank, the binary data shall be forwarded to the app. If not blank, the binary data shall be forwarded to the url with a provided timeout in seconds.
timeout	Integer	False	Optional timeout for HTTP requests Required if a URL is provided
fileType	FileType	False	Optional file type (meant for HTTP file requests).
offset	Integer	False	Optional offset in bytes for resuming partial data chunks
length	Integer	False	Optional length in bytes for resuming partial data chunks

OnHashChange

Message Type: **notification**

Notification containing an updated hashID which can be used over connection cycles (i.e. loss of connection, ignition cycles, etc.). Sent after initial registration and subsequently after any change in the calculated hash of all persisted app data.

PARAMETERS

VALUE	TYPE	MANDATORY	DESCRIPTION
hashID	String	True	Calculated hash ID to be referenced during RegisterAppInterface.

OnWayPointChange

Message Type: **notification**

Notification which provides the entire LocationDetails when there is a change to any waypoints or destination.

PARAMETERS

VALUE	TYPE	MANDATORY	DESCRIPTION
wayPoints	LocationDetails[]	True	See LocationDetails

OnInteriorVehicleData

Message Type: **notification**

PARAMETERS

VALUE	TYPE	MANDATORY	DESCRIPTION
moduleData	ModuleData	True	

OnRCStatus

Message Type: **notification**

Issued by SDL to notify the application about remote control status change on SDL

PARAMETERS

VALUE	TYPE	MANDATORY	DESCRIPTION
allowed	Boolean	False	If "true" - RC is allowed; if "false" - RC is disallowed.
allocatedModules	ModuleData[]	True	Contains a list (zero or more) of module types that are allocated to the application.
freeModules	ModuleData[]	True	Contains a list (zero or more) of module types that are free to access for the application.

OnAppServiceData

Message Type: **notification**

This notification includes the data that is updated from the specific service

PARAMETERS

VALUE	TYPE	MANDATORY	DESCRIPTION
serviceData	AppServiceData	True	

OnSystemCapabilityUpdated

Message Type: **notification**

A notification to inform the connected device that a specific system capability has changed.

PARAMETERS

VALUE	TYPE	MANDATORY	DESCRIPTION
systemCapability	SystemCapability	True	The system capability that has been updated

OnAppCapabilityUpdated

Message Type: **notification**

A notification to inform SDL Core that a specific app capability has changed.

VALUE	TYPE	MANDATORY	DESCRIPTION
appCapability	AppCapability	True	The app capability that has been updated

OnUpdateFile

Message Type: **notification**

This notification tells an app to upload and update a file with a given name.

PARAMETERS

VALUE	TYPE	MANDATORY	DESCRIPTION
fileName	String	True	File reference name.

OnUpdateSubMenu

Message Type: **notification**

This notification tells an app to update the AddSubMenu or its 'sub' AddCommand and AddSubMenus with the requested data

PARAMETERS

VALUE	TYPE	MANDATORY	DESCRIPTION
menuID	Integer	True	This menuID must match a menuID in the current menu structure
updateSubCells	Boolean	False	If not set, assume false. If true, the app should send AddCommands with parentIDs matching the menuID. These AddCommands will then be attached to the submenu and displayed if the submenu is selected.

EncodedSyncPData

Message Type: **request**
Deprecated since: 7.1.0
Allows encoded data in the form of SyncP packets to be sent to the SYNC module. Legacy / v1 Protocol implementation; use SyncPData instead. *DEPRECATED*

PARAMETERS

VALUE	TYPE	MANDATORY	DESCRIPTION
data	String[]	True	Contains base64 encoded string of SyncP packets.

EncodedSyncPData

Message Type: **response**
Deprecated since: 7.1.0

PARAMETERS

VALUE	TYPE	MANDATORY	DESCRIPTION
success	Boolean	True	true, if successful; false, if failed
resultCode	Result	True	See Result
info	String	False	Provides additional human readable info regarding the result.

OnEncodedSyncPData

Message Type: **notification**
Deprecated since: 7.1.0
Callback including encoded data of any SyncP packets that SYNC needs to send back to the mobile device. Legacy / v1 Protocol implementation; responds to EncodedSyncPData.
DEPRECATED

PARAMETERS

VALUE	TYPE	MANDATORY	DESCRIPTION
data	String[]	True	Contains base64 encoded string of SyncP packets.
URL	String	False	If blank, the SyncP data shall be forwarded to the app. If not blank, the SyncP data shall be forwarded to the provided URL.
Timeout	Integer	False	If blank, the SyncP data shall be forwarded to the app. If not blank, the SyncP data shall be forwarded with the provided timeout in seconds.

RPC Spec Generator

The RPC Spec repository contains a parsing script that is used by the following platforms' generator implementations:

- SDL Core
- SDL iOS
- SDL Java Suite
- SDL JavaScript Suite

The RPC Spec repository is imported as a git submodule into each of the previously mentioned platforms. The RPC Spec parser reads the MOBILE_API.xml which is the specification for all RPC messages communicated between a proxy platform and SDL Core. Each platform contains a code generator for creating RPC classes and files. These generators greatly reduce the amount of effort for adding new RPC messages.

Using the RPC Spec Generator

All developers who are using an SDL platform should run these commands in the SDL platform source directory to ensure the latest RPC spec and parser is being used in their builds:

```
git submodule init
git submodule update
```

Using the parser requires a minimum python version, 3.5. The rest of the dependencies for the generators can be installed via:

```
python3 -m pip install -r InterfaceParser/requirements.txt
```

Each platform has their own generator implementation but the instructions for generating their respective RPC code is the same. Refer to the following command line options for running an SDL platform generator script:

```
Mobile API Spec Generator

optional arguments:
  -h, --help          show this help message and exit
  -v, --version       print the version and exit
  -xml SOURCE_XML, --source-xml SOURCE_XML, --input-file SOURCE_XML
                      should point to MOBILE_API.xml
  -xsd SOURCE_XSD, --source-xsd SOURCE_XSD
  -d OUTPUT_DIRECTORY, --output-directory OUTPUT_DIRECTORY
                      define the place where the generated output_directory
                      should be placed
  -r REGEX_PATTERN, --regex-pattern REGEX_PATTERN
                      only elements matched with defined regex pattern will
                      be parsed and generated
  --verbose           display additional details like logs etc
  -e, --enums         only specified elements will be generated, if present
  -s, --structs       only specified elements will be generated, if present
  -m, -f, --functions   only specified elements will be generated, if present
  -y, --overwrite     force overwriting of existing files in
                      output_directory file, ignore confirmation message
  -n, --skip          skip overwriting of existing files in output_directory
                      file, ignore confirmation message
```

RPC Spec Structure

This section describes the structure of the `MOBILE_API.xml` and the elements and attributes that are required for code generation. Refer to the `MOBILE_API.xsd` for the full schema definition.

interface

The root element is the `<interface>` tag. The `<interface>` tag contains any number of `<enum>`, `<struct>` and `<function>` tags.

Example:

```
<interface name="string" version="string" minVersion="string" date="string">
  <!--Zero or more repetitions:-->
  <enum/>
  <!--Zero or more repetitions:-->
  <struct/>
  <!--Zero or more repetitions:-->
  <function/>
</interface>
```

enum

The `<enum>` element contains any number of `<element>` that represents a set of possible values. The `<enum>` has a required `name` attribute.

Example:

```
<enum deprecated="boolean" internal_scope="string" name="string" platform="string"
since="string" until="string">
  <!--Zero or more repetitions:-->
  <description>string</description>
  <!--Zero or more repetitions:-->
 <element hexvalue="string" internal_name="string" name="string"
rootscreen="boolean" since="string" value="integer">
    <!--Zero or more repetitions:-->
    <description>string</description>
    <!--Zero or more repetitions:-->
    <warning>string</warning>
  </element>
  <!--Optional:-->
  <history>
    <enum/>
  </history>
</enum>
```

SDL has two different enum types: `string` and `integer`. If `"hexvalue"` or `"value"` attribute exists, the enum type is `integer`, otherwise the enum type is `string`.

The `<element>` has a required `"name"` attribute. For `string` enums, the value of `"name"` attribute will be one of the possible values of the particular `<enum>`. For `integer` enums, the value of `"value"` attribute will be one of the possible values. The `"hexvalue"`

is just a hexadecimal representation of the `"value"` attribute and either hexvalue or value can be used.

`<element>` could also have an `"internal_name"` attribute. This attribute is not used in communication with SDL Core, but it describes the `<element>` developer facing element value. For example refer to the `HMILevel` enum:

```xml
<enum name="HMILevel" since="1.0">
  <description>Enumeration that describes current levels of HMI.</description>
  <element name="FULL" internal_name="HMI_FULL" />
  <element name="LIMITED" internal_name="HMI_LIMITED" />
  <element name="BACKGROUND" internal_name="HMI_BACKGROUND" />
  <element name="NONE" internal_name="HMI_NONE" />
</enum>
```

The HMILevel communicated to the proxies from SDL Core will use the `name` value (`FULL`, `NONE`, etc). However the `HMILevel` enum usage in SDL Core's code will use the `internal_name` value (`HMI_FULL`, `HMI_NONE`, etc).

struct

`<struct>` is a complex data type. `<struct>` contains any number of `<param>`. The `<struct>` has a required `"name"` attribute.

Example:

```xml
<struct deprecated="boolean" name="string" since="string" until="string">
  <!--Optional:-->
  <history>
    <struct/>
  </history>
  <!--Zero or more repetitions:-->
  <description>string</description>
  <!--Zero or more repetitions:-->
  <param array="boolean" defvalue="integer|decimal|boolean|string"
deprecated="boolean" mandatory="boolean" maxlength="integer" maxsize="integer"
maxvalue="decimal" minlength="integer" minsize="integer" minvalue="decimal"
name="string" since="string" type="string" until="string">
    <!--Zero or more repetitions:-->
    <description>string</description>
    <!--Optional:-->
    <history>
      <param/>
    </history>
  </param>
</struct>
```

Each `<param>` requires `"name"`, `"type"`, and `"mandatory"` attributes. The `"type"` attribute value should be one of `"Boolean"`, `"Float"`, `"Integer"`, `"String"`, or one of the `<enum>`, `<struct>` names specified in the Mobile API.

`<param>` could have `"array"` attribute to represent an array of values or objects of the described type. Attributes `"maxsize"` and `"minsize"` provide additional restrictions to an array.

Numeric types can be restricted using `"minvalue"` and `"maxvalue"`.

The `"defvalue"` attribute contains default value with different type, depends on `<param>` type, note: this attribute is not allowed for `<struct>` type.

function

The `<function>` element represents a specific RPC and message type of the Mobile API. It contains any number of `<param>`. The `<function>` has required `"name"`, `"messagetype"`, and `"functionID"` attributes.

Example:

```xml
<function deprecated="boolean" functionID="string" messagetype="string"
name="string" since="string" until="string">
  <!--Optional:-->
  <history>
    <function/>
  </history>
  <!--Zero or more repetitions:-->
  <description>string</description>
  <!--Zero or more repetitions:-->
  <param array="boolean" defvalue="integer|decimal|boolean|string"
deprecated="boolean" mandatory="boolean" maxlength="integer" maxsize="integer"
maxvalue="decimal" minlength="integer" minsize="integer" minvalue="decimal"
name="string" platform="string" since="string" type="string" until="string">
    <!--Zero or more repetitions:-->
    <description>string</description>
    <!--Optional:-->
    <history>
      <param/>
    </history>
    <!--Zero or more repetitions:-->
    <todo>string</todo>
    <!--Zero or more repetitions:-->
    <element name="string">
      <!--Optional:-->
      <description>string</description>
    </element>
  </param>
</function>
```

The "messagetype" attribute value should be either "request" , "response" , or "notification" .

The "functionID" attribute value should match the "name" attribute of one <element> from the "FunctionID" <enum> .

Just like <param> elements in <struct> , each <param> has required "name" , "type" , and "mandatory" attributes. The "type" attribute value should be one of "Boolean" , "Float" , "Integer" , "String" , or one of the <enum> , <struct> name exists in XML.

<param> could have "array" attribute which means the param represents array of described types. Attributes "max*" and "min*" provide additional restrictions.

The `"defvalue"` attribute contains default value with different type, depends on `<param>` type, note: this attribute is not allowed for `<struct>` type.

RPC Spec Versioning

Any element, param, function, or struct can use the following xml attributes to express in which RPC spec version an item was added, removed, deprecated, or changed.

- `since` - The first version a change was made to an item (added, deprecated, removed)
- `until` - Used with the `since` attribute to determine the range of versions a modification applied to an item was relevant.
- `removed` - A boolean value to describe that an item has been removed from the RPC Spec. Should only be specified if `removed=true`.
- `deprecated` - A boolean value to describe that an item has been deprecated from the RPC Spec. Should only be specified if `deprecated=true`.
- `history` - An array of history items that contains all previous RPC Spec definitions for an item. The most recent version is not included in this history array.

Supported Templates

There are fifteen standard templates to choose from, however some head units may only support a subset of these templates. The following examples show how templates will appear on the Generic HMI and Ford's SYNC 3 HMI.

MEDIA
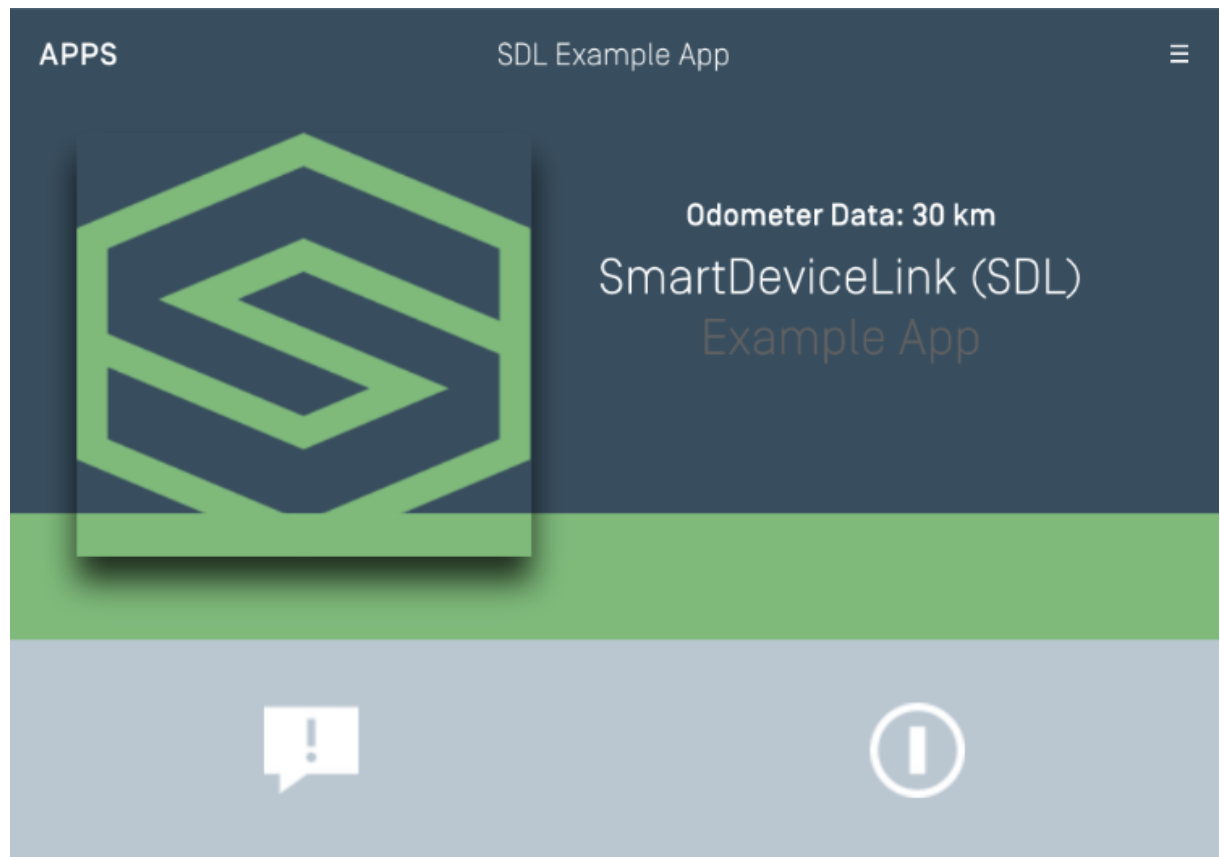
MEDIA (WITH A PROGRESS BAR)

John Prine

Linda Goes to Mars

German Afternoons

00:01:49 / 00:03:06

NON-MEDIA

GRAPHIC WITH TEXT

TEXT WITH GRAPHIC
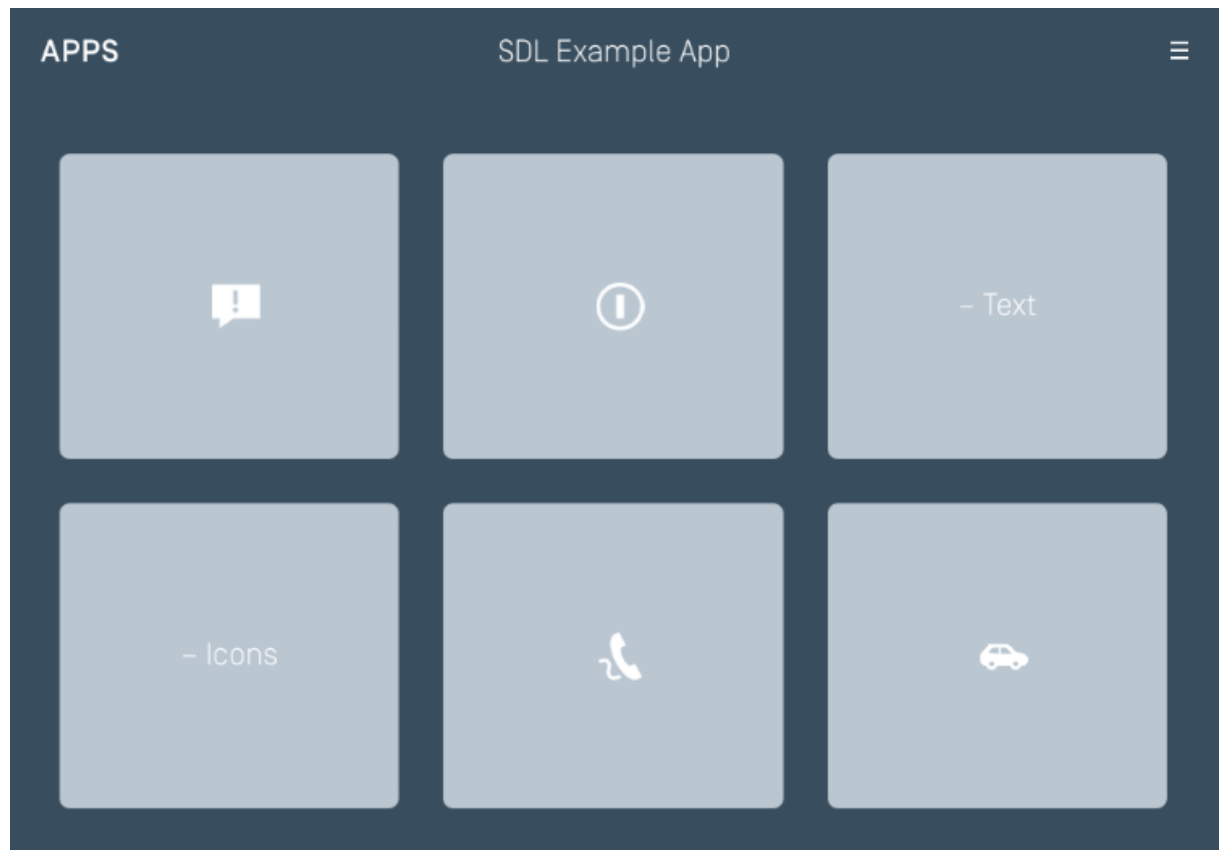
TILES ONLY

GRAPHIC WITH TILES
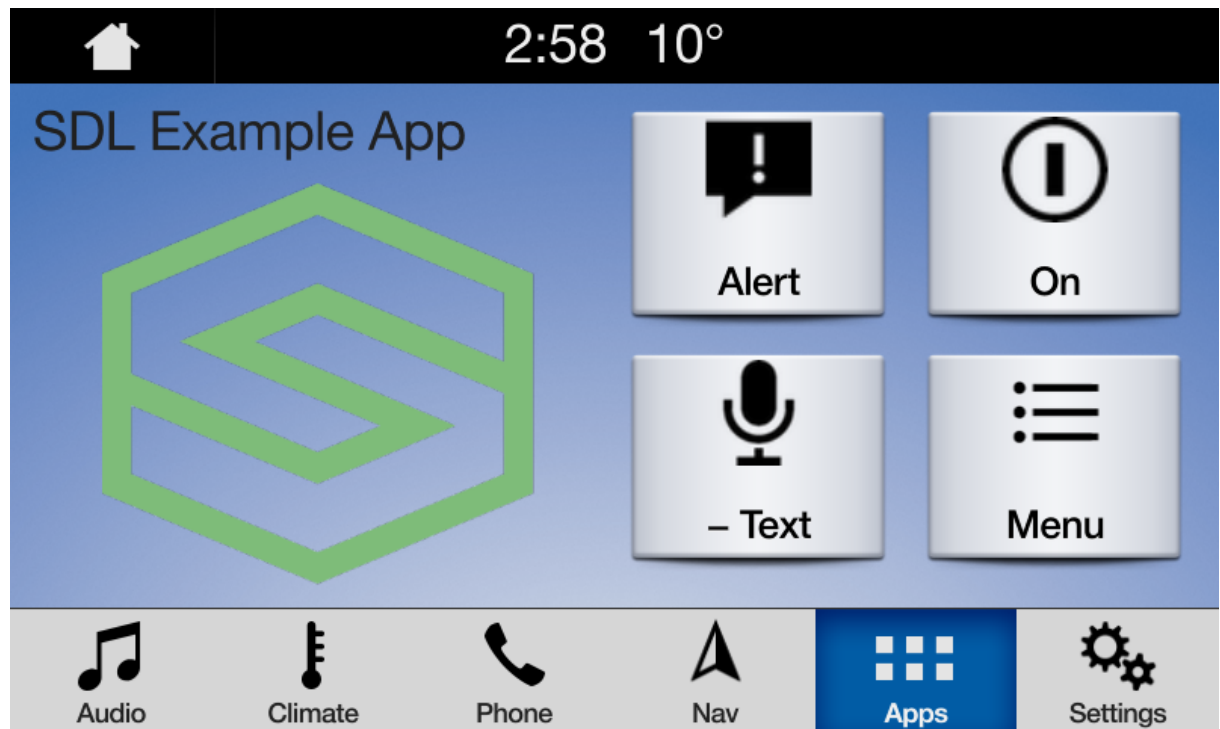
TILES WITH GRAPHIC
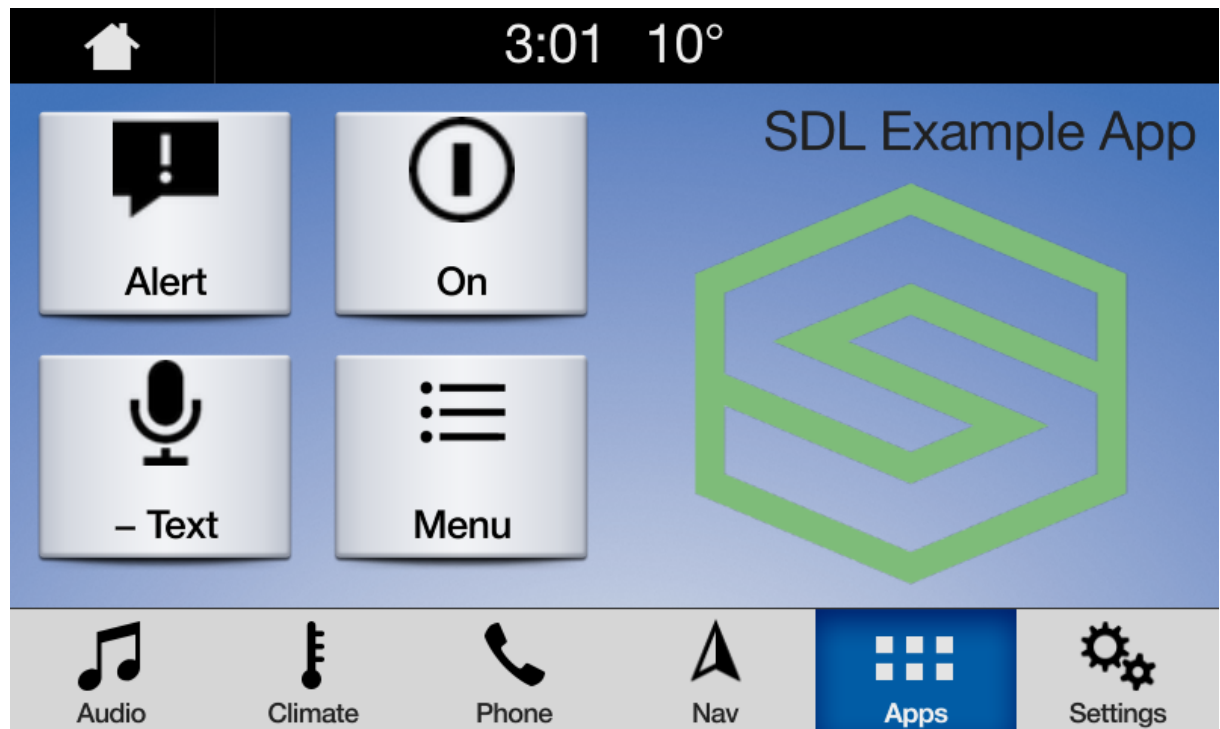
GRAPHIC WITH TEXT AND SOFT BUTTONS
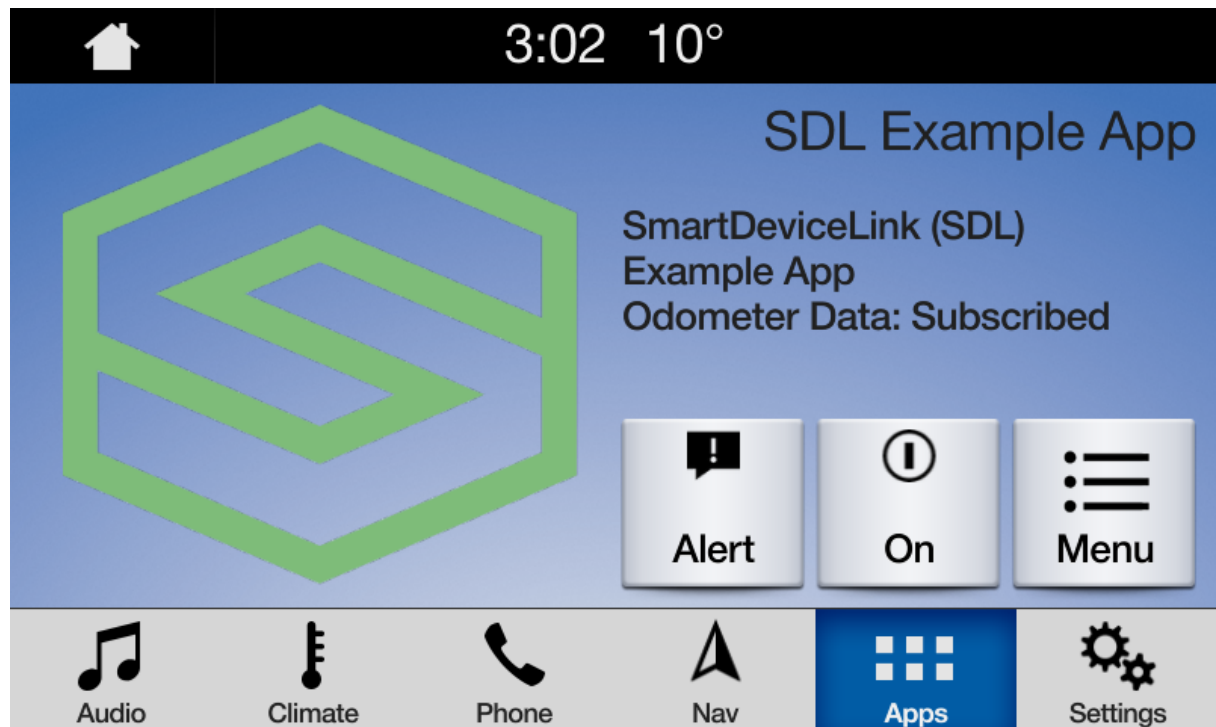
TEXT AND SOFT BUTTONS WITH GRAPHIC
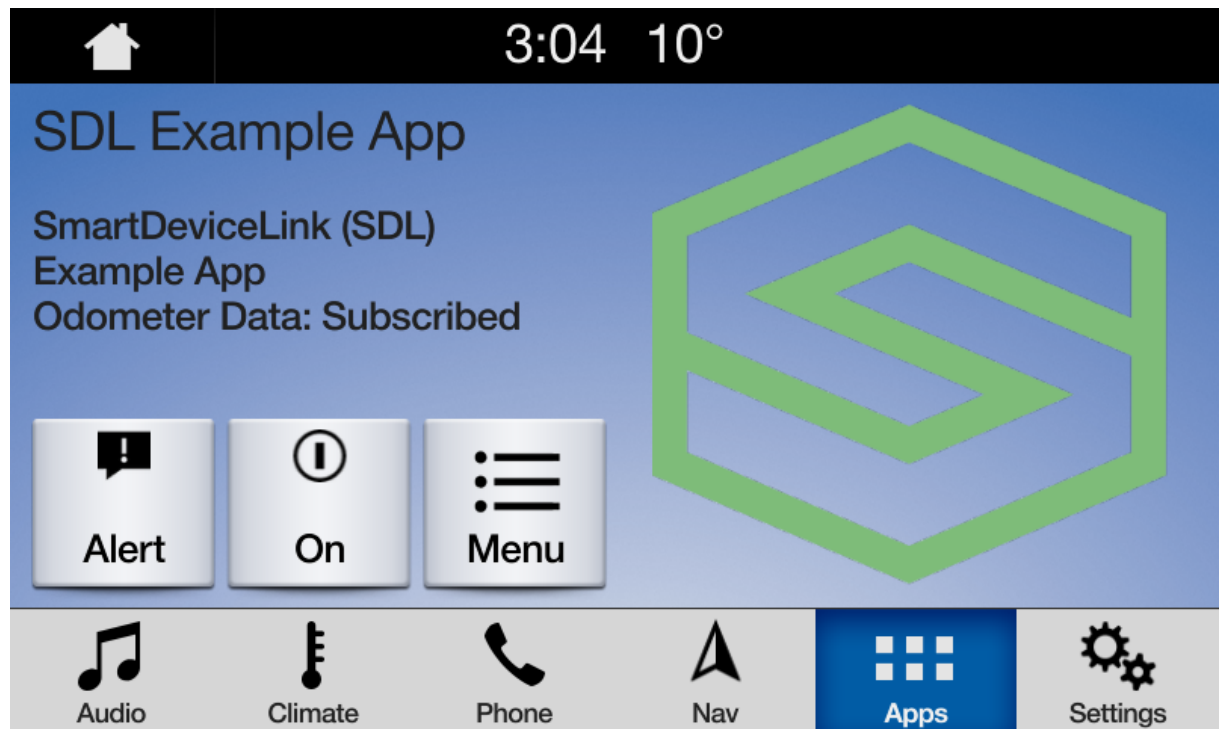
GRAPHIC WITH TEXT BUTTONS
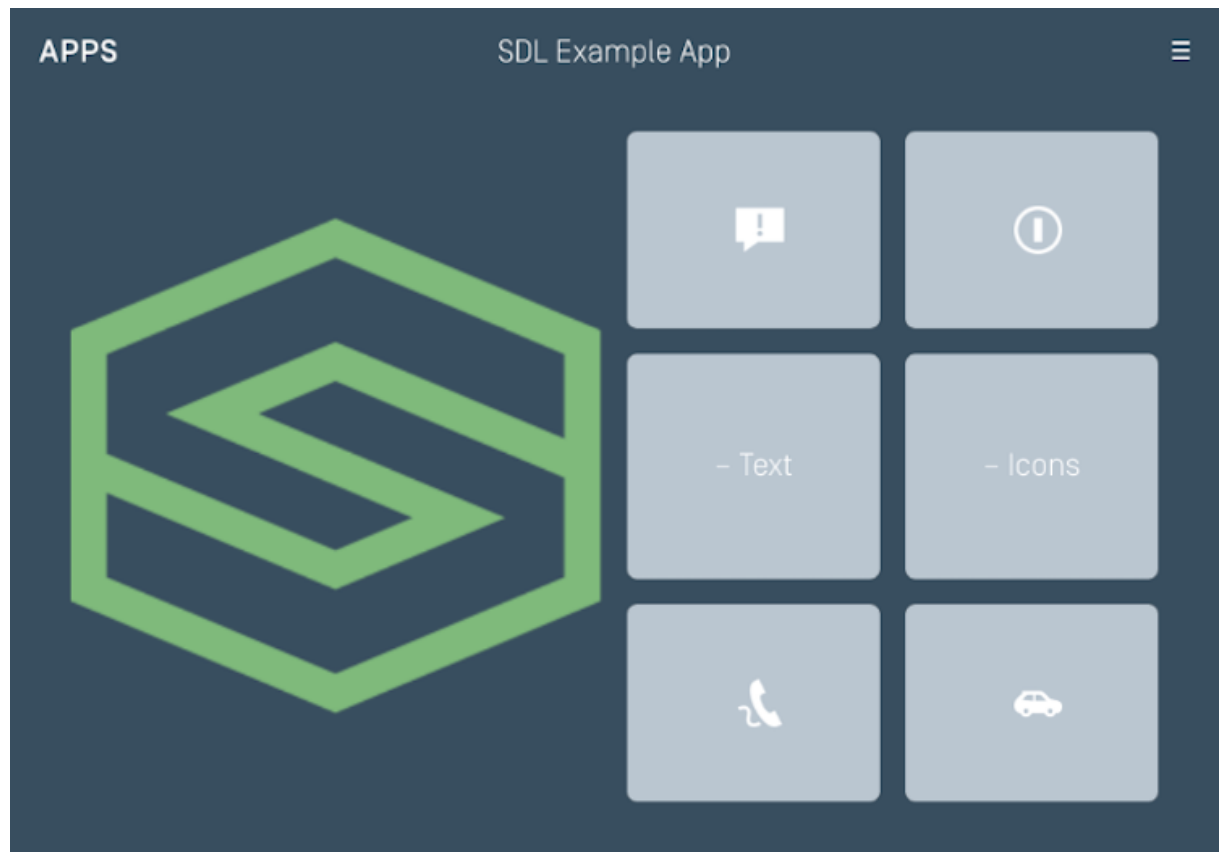
DOUBLE GRAPHIC WITH SOFT BUTTONS
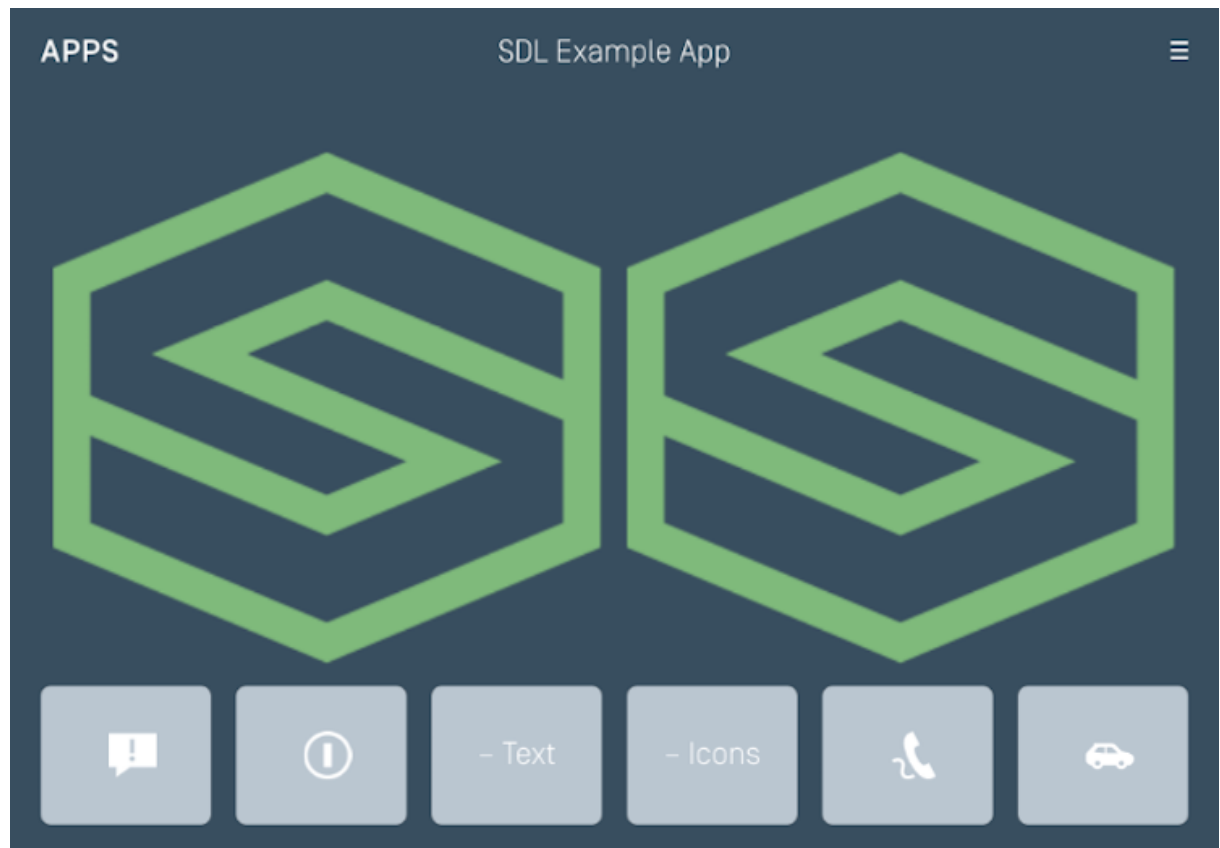
TEXT BUTTONS WITH GRAPHIC
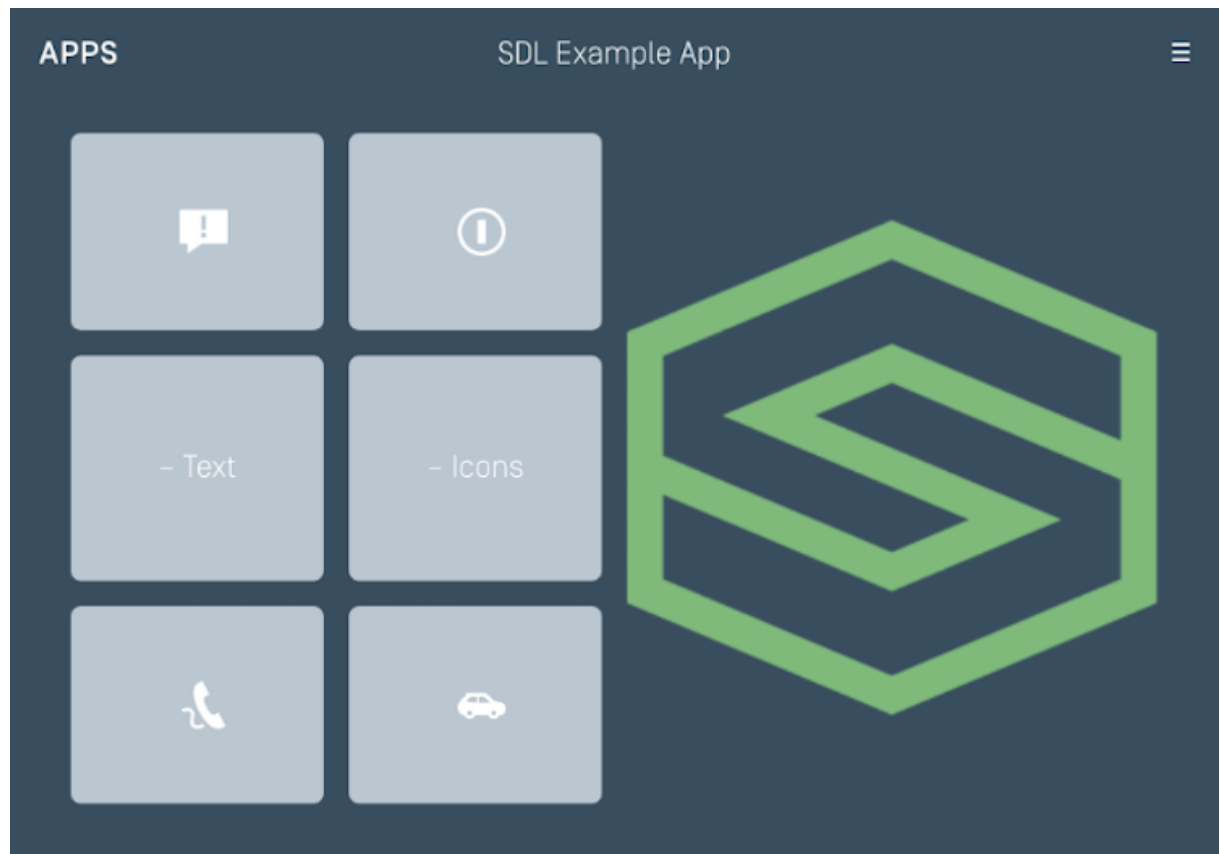
TEXT BUTTONS ONLY
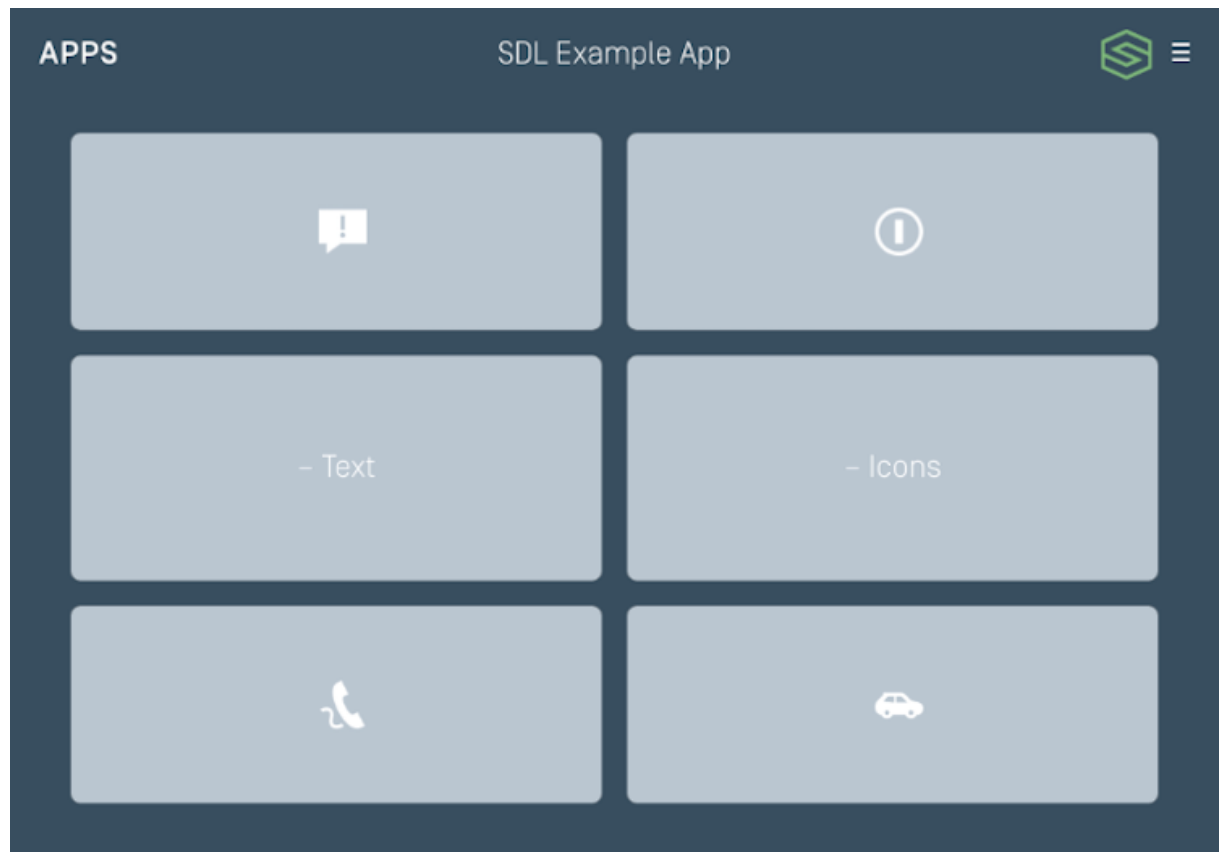
LARGE GRAPHIC WITH SOFT BUTTONS

LARGE GRAPHIC ONLY

Static Icons

Static icons are images that should always be available for applications to use in their integration. These images can be used in any situation where an image is applicable including but not limited to soft buttons, menu items, and choice items.

All Sample images can be downloaded from the Generic HMI project

NAME / DESCRIPTION	HEX VALUE	IMAGE SAMPLE
Accept Call / Active Phone Call / Initiate Phone Call	0x29	
Add Waypoint	0x1B	
Album	0x21	
Ambient Lighting	0x3D	
Arrow - North	0x40	

NAME / DESCRIPTION	HEX VALUE	IMAGE SAMPLE
Audio Mute	0x12	
Audiobook Episode	0x83	
Audiobook Narrator	0x82	
Auxiliary Audio	0x45	
Back / Return	0x86	

NAME / DESCRIPTION	HEX VALUE	IMAGE SAMPLE
Battery Capacity 0 of 5	0xF7	
Battery Capacity 1 of 5	0xF8	
Battery Capacity 2 of 5	0xF9	
Battery Capacity 3 of 5	0xFA	
Battery Capacity 4 of 5	0xF6	

NAME / DESCRIPTION	HEX VALUE	IMAGE SAMPLE
Battery Capacity 5 of 5	0xFB	
Bluetooth Audio Source	0x09	
Bluetooth 1	0xCC	
Bluetooth 2	0xCD	
Browse	0x77	
NAME / DESCRIPTION	HEX VALUE	IMAGE SAMPLE

NAME / DESCRIPTION	HEX VALUE	IMAGE SAMPLE
Cell Phone in Roaming Mode	0x66	
Cell Service Signal Strength 0 of 5 Bars	0x67	
Cell Service Signal Strength 1 of 5 Bars	0x68	
Cell Service Signal Strength 2 of 5 Bars	0x69	
Cell Service Signal Strength 3 of 5 Bars	0x6A	
NAME / DESCRIPTION	HEX VALUE	IMAGE SAMPLE

NAME / DESCRIPTION	HEX VALUE	IMAGE SAMPLE
Cell Service Signal Strength 4 of 5 Bars	0x6B	
Cell Service Signal Strength 5 of 5 Bars	0xD3	
Change Lane - Left	0xC3	
Change Lane - Right	0xC1	
Check Box - Checked	0x27	
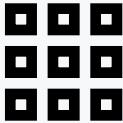
NAME / DESCRIPTION	HEX VALUE	IMAGE SAMPLE
check box - Unchecked	0x28	
Climate	0xD1	
Clock	0xFC	
Compose (eg message)	0x1A	
Contact	0x5C	

NAME / DESCRIPTION	HEX VALUE	IMAGE SAMPLE
Continue	0x42	↑
Dash / Bullet Point	0x7F	—
Date / Calendar	0x87	📅
Delete	0x0F	🗑
Destination	0x94	📍
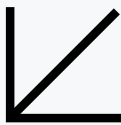
NAME / DESCRIPTION	HEX VALUE	IMAGE SAMPLE
Destination Ferry Ahead	0x4D	
eBookmark (e.g. message, feed)	0x2B	
End Call / reject call	0x2C	
Fail	0xD6	
Fast Forward 30 Seconds	0x08	
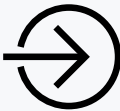
NAME / DESCRIPTION	HEX VALUE	IMAGE SAMPLE
Favorite / Heart	0x0E	
Favorite / Star	0x95	
Fax Number	0x80	
Filename	0x50	
Filter / Search	0x79	
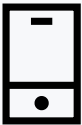
NAME / DESCRIPTION	HEX VALUE	IMAGE SAMPLE
Folder	0x1C	
Fuel Prices	0xE9	
Full Map	0x0C	
Generic Phone Number	0x53	
Genre (Music)	0x4E	

NAME / DESCRIPTION	HEX VALUE	IMAGE SAMPLE
Global Keyboard	0xEA	
Highway Exit Information	0xF4	
Home Phone Number	0x55	
Hyperlink	0x78	
ID3 Tag Unknown	0x51	

NAME / DESCRIPTION	HEX VALUE	IMAGE SAMPLE
Incoming Calls (in list of phone calls)	0x57	
Information	0x5D	
iPod Media Source	0x0D	
Join Calls	0x02	
Keep Left	0x46	

NAME / DESCRIPTION	HEX VALUE	IMAGE SAMPLE

NAME / DESCRIPTION	HEX VALUE	IMAGE SAMPLE
Keep Right	0x48	
Key / Keycode	0x7D	
Left	0x9F	
Left Arrow / Back	0x4B	
Left Exit	0xAF	

NAME / DESCRIPTION	HEX VALUE	IMAGE SAMPLE
Line In Audio Source	0x06	
Locked	0x22	
Media Control - Left Arrow	0x17	
Media Control - Recording	0x20	REC
Media Control - Right Arrow	0x15	

NAME / DESCRIPTION	HEX VALUE	IMAGE SAMPLE
Media Control - Stop	0x16	
Microphone	0xE8	
Missed Calls (in list of phone calls)	0x58	
Mobile Phone Number	0x54	
Move Down / Download	0xE5	

NAME / DESCRIPTION	HEX VALUE	IMAGE SAMPLE
Move Up	0xE4	
MP3 TAG Artist	0x24	
Navigation / Navigation Settings	0x8E	
Navigation Current Direction	0x0A	
Negative Rating - Thumbs Down	0x14	
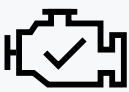
NAME / DESCRIPTION	HEX VALUE	IMAGE SAMPLE
New / Unread Text Message or Email	0x5E	
Office or Work Phone Number	0x56	
Opened / Read Text Message or Email	0x5F	
Origin / Nearby Locale / Current Position	0x96	
Outgoing Calls (in list of phone calls)	0x59	

NAME / DESCRIPTION	HEX VALUE	IMAGE SAMPLE
Phone Call 1	0x1D	
Phone Call 2	0x1E	
Phone Device	0x03	
Phonebook	0x81	
Photo / Picture	0x88	

NAME / DESCRIPTION	HEX VALUE	IMAGE SAMPLE
Play / Pause - pause active	0xD0	▷ ❚❚
Popup	0x76	
Positive Rating - Thumbs Up	0x13	
Power	0x5B	
Primary Phone (Favorite)	0x1F	

NAME / DESCRIPTION	HEX VALUE	IMAGE SAMPLE
Radio Button Checked	0x25	
Radio Button Unchecked	0x26	
Recent Calls / History	0xE7	
Recent Destinations	0xF2	
Redo	0x19	

NAME / DESCRIPTION	HEX VALUE	IMAGE SAMPLE
Refresh	0x97	
Remote Diagnostics - Check Engine	0x7E	
Rendered 911 Assist / Emergency Assistance	0xAC	
Repeat	0xE6	
Repeat Play	0x73	
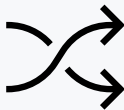
NAME / DESCRIPTION	HEX VALUE	IMAGE SAMPLE
Reply	0x04	
Rewind 30 Seconds	0x07	
Right	0xA3	
Right Exit	0xB1	
Ringtones	0x5A	

NAME / DESCRIPTION	HEX VALUE	IMAGE SAMPLE
Roundabout Left Hand 1	0xEE	
Roundabout Left Hand 2	0x8C	
Roundabout Left Hand 3	0x84	
Roundabout Left Hand 4	0x72	
Roundabout Left Hand 5	0x6E	

NAME / DESCRIPTION	HEX VALUE	IMAGE SAMPLE
Roundabout Left Hand 6	0x64	
Roundabout Left Hand 7	0x60	
Roundabout Right Hand 1	0x62	
Roundabout Right Hand 2	0x6C	
Roundabout Right Hand 3	0x70	

NAME / DESCRIPTION	HEX VALUE	IMAGE SAMPLE
Roundabout Right Hand 4	0x7A	
Roundabout Right Hand 5	0x8A	
Roundabout Right Hand 6	0xEC	
Roundabout Right Hand 7	0xF0	
RSS	0x89	

NAME / DESCRIPTION	HEX VALUE	IMAGE SAMPLE
Settings / Menu	0x49	
Sharp Left	0xA5	
Sharp Right	0xA7	
Show	0xE1	
Shuffle Play	0x74	

NAME / DESCRIPTION	HEX VALUE	IMAGE SAMPLE
Ski Places / Elevation / Altitude	0xAB	
Slight Left	0x9D	
Slight Right	0xA1	
Smartphone	0x05	
Sort List	0x7B	

NAME / DESCRIPTION	HEX VALUE	IMAGE SAMPLE

NAME / DESCRIPTION	HEX VALUE	IMAGE SAMPLE
Speed Dial Numbers - Number 0	0xE0	0
Speed Dial Numbers - Number 1	0xD7	1
Speed Dial Numbers - Number 2	0xD8	2
Speed Dial Numbers - Number 3	0xD9	3
Speed Dial Numbers - Number 4	0xDA	4

NAME / DESCRIPTION	HEX VALUE	IMAGE SAMPLE
Speed Dial Numbers - Number 5	0xDB	5
Speed Dial Numbers - Number 6	0xDC	6
Speed Dial Numbers - Number 7	0xDD	7
Speed Dial Numbers - Number 8	0xDE	8
Speed Dial Numbers - Number 9	0xDF	9
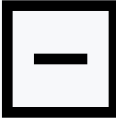
NAME / DESCRIPTION	HEX VALUE	IMAGE SAMPLE
Success / Check	0xD5	
Track Title / Song Title	0x4C	
Traffic Report	0x2A	
Turn List	0x10	
U-Turn Left Traffic	0xAD	

NAME / DESCRIPTION	HEX VALUE	IMAGE SAMPLE
U-Turn Right Traffic	0xA9	
Undo	0x18	
Unlocked	0x23	
USB Media Audio Source	0x0B	
Voice Control Scrollbar - List Item number 1	0xC7	1

NAME / DESCRIPTION	HEX VALUE	IMAGE SAMPLE
Voice Control Scrollbar - List Item number 2	0xC8	2
Voice Control Scrollbar - List Item number 3	0xC9	3
Voice Control Scrollbar - List Item number 4	0xCA	4
Voice Recognition - Failed	0x90	
Voice Recognition - Pause	0x92	

NAME / DESCRIPTION	HEX VALUE	IMAGE SAMPLE
Voice Recognition - Successful	0x8F	
Voice Recognition - System Active	0x11	
Voice Recognition - System Listening	0x91	
Voice Recognition - Try Again	0x93	
Warning / Safety Alert	0xFE	

NAME / DESCRIPTION	HEX VALUE	IMAGE SAMPLE
Weather	0xEB	
WiFi Full	0x43	
Zoom In	0x98	
Zoom Out	0x9A	

Policies

SmartDeviceLink uses a robust message checking system that allows OEMs to control exactly what information and RPCs can be sent to and from any connected app. This allows the OEM to be in control of the user's data and make the choices of which apps get

access to it. The central feature of policies is to disallow certain RPC requests from the application when they are not permitted while also allowing other apps to use those RPCs. It can also be as granular as specifying which parameters can be accepted for messages, and as broad as disallowing an app from registering in the first place. This is very useful when sensitive data like vehicle data is involved because policies can specify exactly which data sets (RPM, PRNDL, etc) apps can access.

Why is this type of control useful? Let's examine an app's ability to use the `Alert` RPC. Often an OEM will wish to choose which apps have the ability to show an Alert overlay on screen while the app is in the background or when the app has not been activated by the user.

While this could be a simple check hardcoded into Core that disallowed all apps from showing Alerts while backgrounded, it may also make sense that some apps are allowed to do so. By creating different functional groups and assigning those to specific apps, Core can have dynamic behavior for each app.

Common Use Cases for Policies

- Permitting and denying access to vehicle data.
- Permitting and denying access to remote control of vehicle features like the climate system.
- Adjusting the number of messages that the app can send before it is activated.
- Adjusting the default HMI status for apps after registration.
- Including SDL related strings and translations that will be displayed to the user.

Policy Table Updates

Beyond simply just hardcoding behaviors, it is expected that app specific behaviors will not remain static, therefore policy tables are designed to be updated. This process can happen a few different ways, but the most common is leveraging the connected application to retrieve and transfer the policy table through to Core.

SDL provides an example policy server that can dynamically build and serve these policies to head units in order to update their existing policies. We strongly recommend that OEMs support policy table updates using a policy server so that apps which become available after a head unit ships can have specialized policies.
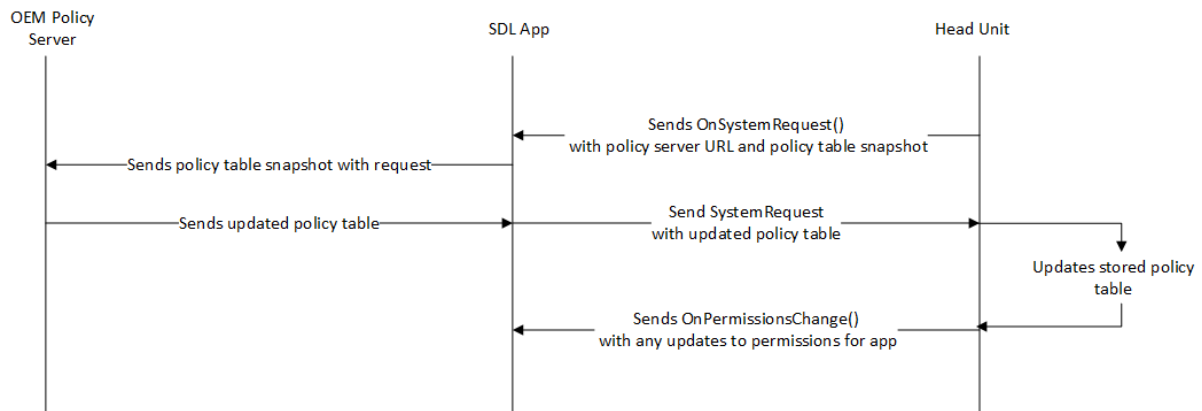
POLICY TABLE UPDATE TRIGGERS

Some events will trigger Core to request an updated policy table. See the Policy Table Update Configurations section for more information on how these triggers work. Examples include:

- When a new and unrecognized app connects.
- When a new device is first given SDL permissions.
- When a configurable number of ignition cycles have occurred since the last update.

Policy Table Fields

Module Config

The module config section contains some global defaults that can be set for SDL.

Policy Table Update Configurations

```json
"module_config": {
    ...
    "exchange_after_x_ignition_cycles": 100,
    "exchange_after_x_kilometers": 1800,
    "exchange_after_x_days": 30,
    "timeout_after_x_seconds": 60,
    "seconds_between_retries": [
        1,
        5,
        25,
        125,
        625
    ],
    ...
}
```

A Policy Table Update (PTU) is when SDL sends a request to a connected application to retrieve a new policy table from the server.

Some configurable events will trigger Core to request a PTU. The exact parameters which are used to trigger the update can be configured in this section:

NAME	TYPE	DESCRIPTION
exchange_after_x_ignition_cycles	Integer	PTU Trigger, defines the number of ignition cycles before SDL initiates a PTU
exchange_after_x_kilometers	Integer	PTU Trigger, defines the distance that can be traveled in the vehicle before SDL initiates a PTU
exchange_after_x_days	Integer	PTU Trigger, defines the number of days that can pass before SDL initiates a PTU
timeout_after_x_seconds	Integer	The amount of time (in seconds) SDL will wait for a PTU to complete before timing out and retrying
seconds_between_retries	Integer Array	A list of times (in seconds) to wait after a failed PTU before trying again. The number of items in this list determines the number of retries that will be attempted in a single session.

lock_screen_dismissal_enabled

A boolean value which determines if it should be possible to dismiss the lock screen on a connected SDL application.

If true, an entry must be present in the `consumer_friendly_messages` section of the policy for `LockScreenDismissalWarning`. The `textBody` section of this structure will be displayed to the user in this case, warning them that dismissal should only be performed if

they are not the driver of the vehicle. If the entry is not present, then this option will be ignored and default to `false` .

endpoints

This section contains several lists of urls that are used throughout SDL. Each subsection must contain a default list of endpoints, and app-specific endpoints can be added optionally as well. If an app-specific endpoint list is specified, these endpoints will be used instead of the default when performing the corresponding request with that application.

EXAMPLE ENTRY

```
"module_config": {
    ...
    "endpoints": {
        "0x07": {
            "default": [
                "http://x.x.x.x:3000/api/1/policies/proprietary"
            ],
            "123abc": [
                "https://app.endpoint.com/api/1/policies/proprietary"
            ]
        },
        "0x04": {
            "default": [
                "http://x.x.x.x:3000/api/1/softwareUpdate"
            ],
            "123abc": [
                "https://example.endpoint.com/api/1/softwareUpdate"
            ]
        },
        "queryAppsUrl": {
            "default": [
                "http://sdl.shaid.server"
            ]
        },
        "lock_screen_icon_url": {
            "default": [
                "http://i.imgur.com/TgkvOIZ.png"
            ],
            "123abc": [
                "https://example.endpoint.com/logo.png"
            ]
        },
        "custom_vehicle_data_mapping_url": {
            "default": [
                "http://x.x.x.x:3000/api/1/vehicleDataMap"
            ]
        }
    },
    ...
}
```

- `0x07` - A set of urls that are used when performing a PTU
- `0x04` - A set of urls that are used to retrieve module software updates
- `queryAppsUrl` - A set of urls that are used to receive valid apps for querying on iOS devices
- `lock_screen_icon_url` - A set of urls which each host an image that can be displayed by the application on the driver's device while the lock screen is active (ex. an OEM

logo)

- This url is sent in a request to each application after they are first registered. The application proxy downloads the image and displays it along with the lock screen when driver distraction mode is active.

- `custom_vehicle_data_mapping_url` - A set of urls that are used to retrieve a custom mapping file to be used by the HMI to translate requests for custom vehicle data (as defined in the `vehicle_data` section)

  - See this guide for more information

endpoint_properties

This section includes properties corresponding to the keys in the `endpoints` section:

- `version` - The version of the file associated with this url, currently only applies to `custom_vehicle_data_mapping_url`

EXAMPLE ENTRY

```
"module_config": {
    ...
    "endpoint_properties": {
        "custom_vehicle_data_mapping_url": {
            "version": "1.0.0"
        }
    },
    ...
}
```

notifications_per_minute_by_priority

This section includes a list of custom `priority` values to be used in the `app_policies` section. Each key is attached to a value in this map, which defines how many alerts can be sent by an app while in the background ("notifications") within a given minute (ex. `"NAVIG`

`ATION": 15` means that an application with `"priority": "NAVIGATION"` can send up to 15 background alerts every minute).

This feature is used to restrict an app's control of the screen when they are in the background, so that they are not disruptive to the user while driving.

EXAMPLE ENTRY

```
"module_config": {
    ...
    "notifications_per_minute_by_priority": {
        "EMERGENCY": 60,
        "NAVIGATION": 15,
        "PROJECTION": 15,
        "VOICECOM": 20,
        "COMMUNICATION": 6,
        "NORMAL": 4,
        "NONE": 0
    },
    ...
}
```

certificate

A field used to replace the local certificate used for establishing secure connections with applications. The value of this field is the full string representation of the certificate. If present in a PTU, this certificate will overwrite the existing certificate on the module.

Consumer Friendly Messages

The consumer friendly messages section contains a list of messages which are meant to be displayed by the HMI at specified times. Most can be custom-defined and referenced via the `user_consent_prompt` section of a functional group.

languages

Each `consumer_friendly_messages` entry contains a list of languages that it supports, with each language entry containing a struct with the following possible fields: `tts` , `label` , `line1` , `line2` , and `textBody` .
The HMI can retrieve this struct by key and language via the `SDL.GetUserFriendlyMessage` RPC.

Functional Groupings

The functional groupings section contains the definitions for each named group of RPC permissions that an application can be assigned. There can be any number of functional groups. The functional groups are used in the application policies section to define permissions for each application.

EXAMPLE ENTRY

```json
"functional_groupings" : {
    "Location-1": {
        "user_consent_prompt": "Location",
        "encryption_required": false,
        "rpcs": {
            "GetVehicleData": {
                "hmi_levels": [
                    "BACKGROUND",
                    "FULL",
                    "LIMITED"
                ],
                "parameters": [
                    "gps",
                    "speed"
                ]
            },
            "OnVehicleData": {
                "hmi_levels": [
                    "BACKGROUND",
                    "FULL",
                    "LIMITED"
                ],
                "parameters": [
                    "gps",
                    "speed"
                ]
            },
            "SubscribeVehicleData": {
                "hmi_levels": [
                    "BACKGROUND",
                    "FULL",
                    "LIMITED"
                ],
                "parameters": [
                    "gps",
                    "speed"
                ]
            },
            "UnsubscribeVehicleData": {
                "hmi_levels": [
                    "BACKGROUND",
                    "FULL",
                    "LIMITED"
                ],
                "parameters": [
                    "gps",
                    "speed"
                ]
            }
        }
    },
```

```
      ...
  }
```

encryption_required

This flag is used to specify whether the RPCs included in this functional group must be encrypted when assigned to an application. This flag can be overridden for a specific application by setting the same flag to `false` in its application policies entry (See App Policies).

user_consent_prompt

This field only applies when Core is built with `EXTENDED_POLICY=EXTERNAL_PROPRIETARY`, using this field means that the user will need to provide consent before an application can gain permission to use this group. The consent prompt given to the user will be retrieved from the `consumer_friendly_messages` entry tied to this value (ex. if the value is `AppPermissions`, the prompt will be retrieved from the `consumer_friendly_messages.messages.AppPermissions` field of the policy table).

rpcs

This section defines a list of RPCs which will be granted to each application that has access to this functional group. Within each entry are several fields to fine-tune permissions by HMI level or individual parameter:

- `hmi_levels` - A list of HMI levels from which this RPC can be sent by an application if given permissions for this functional group.
- `parameters` - A list of parameters which can be sent in this RPC by an application if given permissions for this functional group. This field is primarily used for the purpose of controlling vehicle data permissions.

  - If an application sends a parameter that it does not have access to, it will be filtered out, and Core will process the message normally from there.
  - If omitted, all parameters will be allowed.

Vehicle Data

This section can be used to define custom vehicle data values for a system, as well as add capabilities for newer vehicle data items on an older module without updating the system software. See this guide for more details.

App Policies

The app policies section defines permissions that each application is granted in the SDL environment. This is where you would change the default permissions for each application, or add rules for specific applications. Each key in this object is associated with the `fullAppID` field for an application (or `appID` if `UseFullAppID = false` in `smartDeviceLink.ini`), and each value is a struct which contains several fields to control permissions for this app. In addition, a few reserved keys exist for this section:

- `default` - The set of permissions which are assigned to an app if it does not have its own entry in the app policies section.
- `pre_DataConsent` - The set of permissions which are assigned to an app if the device is not given SDL permissions by the user. Only applies when Core is built with `EXTERNAL_PROPRIETARY` policy mode.

Basic Fields

Example Entry

```
"app_policies": {
    "123abc": {
        "keep_context": false,
        "steal_focus": false,
        "priority": "NONE",
        "default_hmi": "NONE",
        "groups": ["Base-4"],
        "RequestType": [],
        "RequestSubType": [],
        "nicknames": ["Example Application"],
        "AppHMIType": ["NAVIGATION", "MEDIA"]
    },
    ...
}
```

groups

This field defines a list of functional groups which should be made available to this application, populated with a list of keys from the `functional_groupings` section (ex. if this list contains "Base-4", all permissions defined in the "Base-4" functional group will be assigned to this application).

preconsented_groups

This field defines additional groups which are assigned to this application, with the difference that the user is expected to have consented to using these permissions separately from the HMI (via a mobile popup, etc.). Any groups defined in this list do not require consent from the user, even if they have a `user_consent_prompt` defined. Only applies when Core is built with `EXTERNAL_PROPRIETARY` policy mode, otherwise these permissions are simply added to the normal `groups` list.

nicknames

This field defines a list of potential appNames which an application is allowed to register with. If the appName field in the app's `RegisterAppInterface` does not match one of the

values in this list, the message will be rejected. If omitted, no restrictions will be applied to appName.

> **NOTE**
>
> This field is mandatory when defining policies for a cloud application, in which case the first nickname will be used in the application list prior to app registration.

AppHMIType

This field defines a list of AppHMITypes which an application is allowed to be registered with. If provided, this list of AppHMITypes will be used for this application in place of the `appHMIType` list provided in the app's RegisterAppInterface request. If there are AppHMITypes present in the app's RegisterAppInterface request that are not present in this list, SDL Core will provide a warning of this discrepancy to the app in the RegisterAppInterface response.

> **NOTE**
>
> This field is mandatory for webengine apps which use the `WEB_VIEW` AppHMIType, in which case the `WEB_VIEW` value must be explicitly included in this list.

keep_context

This field is a flag which allows or disallows an application to use soft buttons with the `KEEP_CONTEXT` action. This field only applies when Core is built with `EXTERNAL_PROPRIETARY` policy mode, otherwise this permission is always allowed.

steal_focus

This field is a flag which allows or disallows an application to use soft buttons with the `STEAL_FOCUS` action. This field only applies when Core is built with `EXTERNAL_PROPRIETARY` policy mode, otherwise this permission is always allowed.

priority

String value tied to the `notifications_per_minute_by_priority` section of `module_config`. By assigning a priority value from this list, the application is allowed the number of background alerts which is specified by that category (see notifications_per_minute_by_priority).

default_hmi

This field defines the default HMI level that the application will be initialized to when it is registered for the first time. This field only applies when Core is built with `EXTERNAL_PROPRIETARY` policy mode, otherwise the default HMI level is always `NONE`.

RequestType

This field defines a list of possible values for `requestType` that can be used in `OnSystemRequest` and `SystemRequest` RPCs when communicating with this app.

RequestSubType

This field defines a list of possible values for `requestSubType` that can be used in `OnSystemRequest` and `SystemRequest` RPCs when communicating with this app.

heart_beat_timeout_ms

Heartbeat timeout for this application, deprecated in Protocol Version 4, which was introduced in SDL Core Version 4.0.0.

Security Fields

Example Entry

```
"app_policies": {
    "123abc": {
        "keep_context": false,
        "steal_focus": false,
        "priority": "NONE",
        "default_hmi": "NONE",
        "groups": ["Base-4"],
        "RequestType": [],
        "RequestSubType": [],
        "certificate": "-----BEGIN CERTIFICATE-----\n...",
        "encryption_required": true
    },
    ...
}
```

certificate

This field is a string value containing a certificate for establishing a secure connection with the application. This certificate is currently only used for cloud applications when establishing a secure WebSocket connection.

encryption_required

This flag can be used to override `encryption_required` on the functional group level. If `false`, then this application will not be required to encrypt any of the RPCs which it has permissions to, even if the associated functional group has `encryption_required` set to `true`. If omitted, this flag defaults to `true`.

Remote Control Fields

Example Entry

```
"app_policies": {
  "123abc": {
    "keep_context": false,
    "steal_focus": false,
    "priority": "NONE",
    "default_hmi": "NONE",
    "groups": ["Base-4", "RemoteControl"],
    "RequestType": [],
    "RequestSubType": [],
    "AppHMIType": ["REMOTE_CONTROL"],
    "moduleType": ["RADIO", "CLIMATE"]
  },
  ...
}
```

moduleType

This field defines a list of Remote Control module types (defined in the `ModuleType`
enum in the Mobile API) which are accessible by the app. If empty, all module types will be
allowed. If omitted, no module types will be allowed.

App Service Fields

These fields are used if the application is either a provider or consumer of app service
data. See our App Service Guidelines for more information on developing an application

with these capabilities.

Example Entry

```
"app_policies": {
  "123abc": {
    "keep_context": false,
    "steal_focus": false,
    "priority": "NONE",
    "default_hmi": "NONE",
    "groups": ["Base-4", "AppServiceProvider", "AppServiceConsumer"],
    "RequestType": [],
    "RequestSubType": [],
    "app_services": {
      "MEDIA": {
        "service_names": ["SDL Music", "App Music"],
        "handled_rpcs": [{"function_id": 41}]
      },
      "NAVIGATION": {
        "service_names": ["SDL Navigation", "App Navigation"],
        "handled_rpcs": [{"function_id": 39}]
      },
      "WEATHER": {
        "service_names": ["SDL Weather", "App Weather"],
        "handled_rpcs": []
      }
    },
    "allow_unknown_rpc_passthrough": true
  },
  ...
}
```

app_services

This field defines the properties of the app services that this application is allowed to publish. Each key within this object corresponds to an app service type (ex. "MEDIA").

- `service_names` - Defines a list of potential service names that an app can publish with this service type.

  - If the `serviceName` field in the app's `PublishAppService.appServiceManifest` does not match one of the values in this list, the message will be rejected.
  - If omitted, no restrictions will be applied to the `serviceName` parameter.

- `handled_rpcs` - This field defines a list of permissions for RPCs that an app service could intercept via the RPC passing feature.

    - If the `handledRPCs` field in the app's `PublishAppService.appServiceManifest` includes a function ID that is not in this list, the message will be rejected.
    - This field is mandatory, but can be empty.
    - Subfields:

        - `function_id` - The function ID of the potential handled RPC. Can be a function ID which was introduced after Core's local RPC spec version.

allow_unknown_rpc_passthrough

This flag can be used to allow an application to pass RPCs which are unknown to Core (functions which were defined after Core's local API version) to an app service provider. Otherwise, such messages from this application will be rejected. If omitted, this flag defaults to `false`.

Cloud Application Fields

These fields are used for applications which use a cloud transport and require Core to initiate the connection. Each of these fields can be modified by a permitted app via the `SetCloudAppProperties` RPC.

Example Entry

```
"app_policies": {
  "123abc": {
    "keep_context": false,
    "steal_focus": false,
    "priority": "NONE",
    "default_hmi": "NONE",
    "groups": ["Base-4", "CloudApp"],
    "RequestType": [],
    "RequestSubType": [],
    "nicknames": ["Example Application"],
    "endpoint": "ws://smartdevicelink.com:8888/path/",
    "enabled": true,
    "cloud_transport_type": "WS",
    "auth_token": "1234567890",
    "icon_url": "http://i.imgur.com/TgkvOIZ.png"
  },
  ...
}
```

enabled

This flag is used to determine if the cloud application should appear in the app list in the HMI. If `true`, the application will appear in the app list as a cloud application. Selecting this application from the list will cause Core to connect to and register the application.

hybrid_app_preference

This option is used to determine the default behavior if the same app is available via a mobile device and over the cloud (determined by a duplicate `appName`). The possible values for this option are `MOBILE`, `CLOUD`, or `BOTH`.

- If this value is set to `MOBILE`, the cloud application will be unregistered and removed from the app list if both applications are available.
- If this value is set to `CLOUD`, the mobile application will be unregistered if both applications are available.
- If this value is set to `BOTH`, both applications will remain in the app list if they are available.

endpoint

This field defines the endpoint which Core will attempt to connect to when this application is selected from the app list (ex. `ws://smartdevicelink.com:8888/path/`). The format is dependent on the value of `cloud_transport_type` .

auth_token

This field defines an authentication token to identify the user account tied to this vehicle. The format of this string is app-defined, and this value will be sent in the initial StartServiceACK when a connection is established with the cloud application.

cloud_transport_type

This field is a string which defines the type of transport that is used by the cloud application. The value of this is an arbitrary string, since an OEM can implement their transport adapter to establish any type of cloud connection. The current values which are supported by the provided cloud adapter in the SDL Core project are `WS` and `WSS` .

icon_url

This field is used to point to an app icon which can be displayed for a cloud application before it is connected. When provided, Core will retrieve this image using a connected application via an `OnSystemRequest` notification with the `ICON_URL` type.

App Services

RELATED EVOLUTION PROPOSALS

- 0167 App Services
- 0223 Add Currently Playing Media Image to MediaServiceData
- 0225 Update Published App Services
- 0239 Media Service Data Progress Bar Improvements

Overview

App Services are a way for apps to offer augmented services to both the onboard embedded system and also to other SDL connected apps. The data sent using this feature is specific based on the app service type. This gives the app service consumer the necessary context for them to build experiences using that data. It also allows them to be prepared to work with many different app providers of the same app service type without having to create unique integrations with each of their services.

Available App Service Types

The defined and supported app types as of writing this guide are:

- Media
- Weather
- Navigation

For the up-to-date list, the `AppServiceType` enum in the RPC spec can be used as reference.

Future App Service Types

The current list of app services is by no means the only services that will ever be available. These services are just the first round of types to be added for their familiarity and usefulness in terms of creating an in-car user experience. As new services become available head units with old versions of SDL Core can still manage the publishing and subscription of those services using the new types. While the module will not be able to make use of the new service, other SDL connected applications can access them as they become defined in the RPC spec and are made available in apps.

App Service Providers

An app service provider is the originator of a service that provides updates of related service data. For example, a connected music player application can be a Media Service or an embedded Navigation app can be a Navigation Service. Therefore an app service provider can live on either side of the SDL connection, core or application library side.

It is up to each app to decide if they wish to expose their service or not. It is not a requirement for apps to do so.

Request an Action be Taken by an App Services Provider

There are a few ways in which an app service provider may grant control over its features. The first is through regular SDL RPCs. The SDL Core project has the ability pass certain RPCs to app service providers when they make sense in terms of the app service type's context. For example, the `SendLocation` RPC is used by the navigation system to add as a destination or waypoint. If a different app other than the embedded navigation system is acting as the active app service, that RPC should be routed there. This allows users to bring in their apps that take the place of built-in services, but allow other apps who interact with those system to continue using their same RPCs and expect similar functionality. This type of action happens passively to both consumer and producers.

The second method to access control of the app service provider is through the `Perform AppServiceInteraction` RPC. This RPC will require the app consumer to have a closer relationship to the app service provider as it will use specific URIs that the provider recognizes. It enables providers to expose a greater feature set to app service consumers without the need of SDL RPC spec modifications and updated SDL Core implementations. App service providers are not required to implement this feature, nor are they required to honor a request. They are expected to reply to the request though. See the RPC spec for more information.

App Service Consumers

An app service consumer retrieves and can subscribe to updates of service data that is provided by an app service provider. For example, the module side with SDL Core could use the media app service data from a music player app to place that information outside the standard SDL "app screen" templates.

A connected SDL app could retrieve navigation app service data from the embedded navigation app service to determine the best place to fill up the car with gas and then display that information in their SDL "app screen."

Consuming App Service Data on the Module

The main use case for using app service data on the module as a consumer is to be able to present that data in a specific and predictable way. The module can maintain branding and their user experience by taking the service data and placing it outside the generic "app screen" template boxes. For example, they may populate the app service data into the places they usually populate their embedded data.

EXAMPLE WITH GENERIC HMI

The following is an example of populating specific areas of the HMI with app service data.

COLLAPSED VIEW

This view contains very brief info from the app service including the next navigation instruction icon from the navigation service provider, the album art of the currently playing song from a media service provider, and the current outside temperature from the weather service provider. For this user experience, tapping on a certain tile will expand that app service, and a similar gesture like double tap, long press on a tile, or tapping on the up arrow icon will explode the view into an overlay that displays more of all the service data.
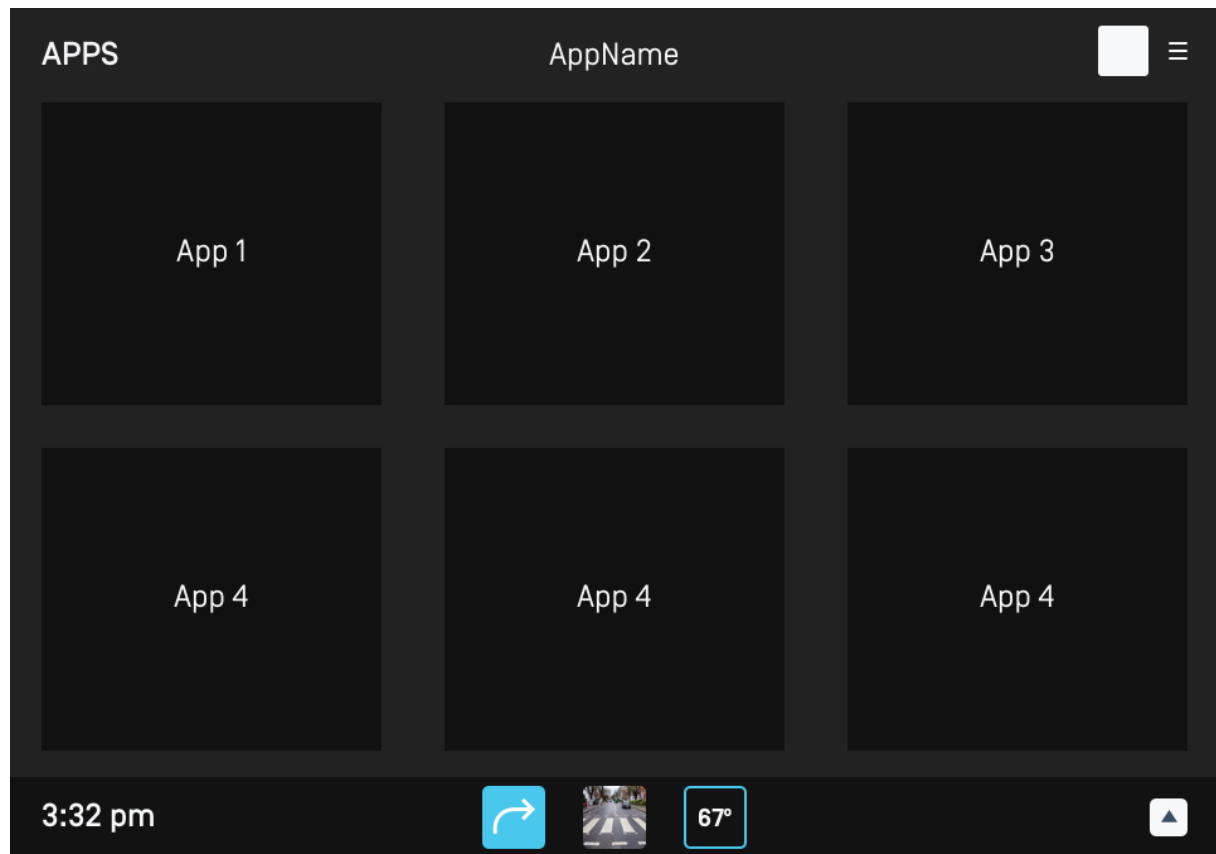
NAVIGATION EXPANDED

The expanded view for the navigation tile would include more specific data on the next navigation instruction including street name, the instruction type that must be take (both from the head of the `instructions` array), and the distance until that instruction must take place (`nextInstructionDistance`).

APPS   AppName   ☐ ☰

App 1	App 2	App 3
App 4	App 4	App 4

3:32 pm   ↱ **Lincoln Ave** Turn Right | .5 mi   67° ▲
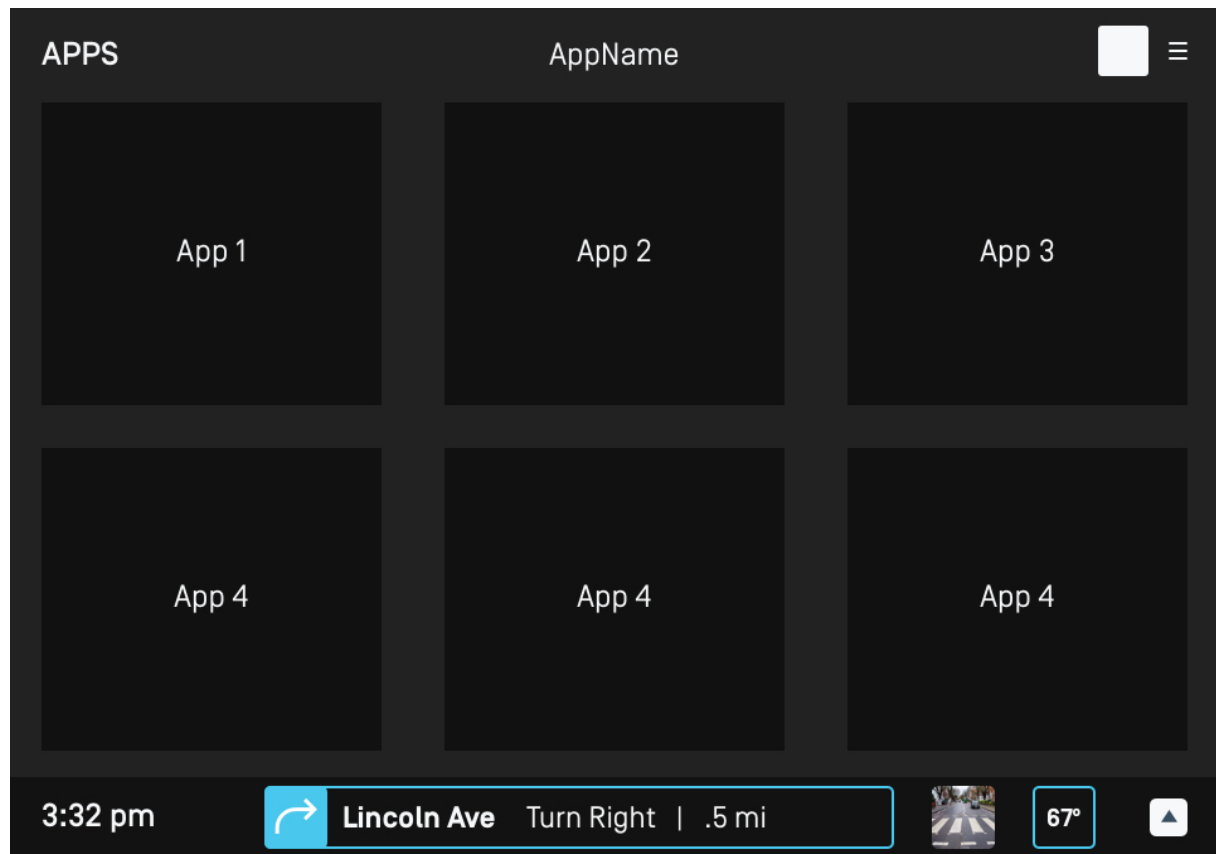
WEATHER EXPANDED

The expanded weather view provides more information on the current conditions including the temperatureHigh / temperatureLow for the day, the chance of precipitation (precipProbability), and the current windSpeed .
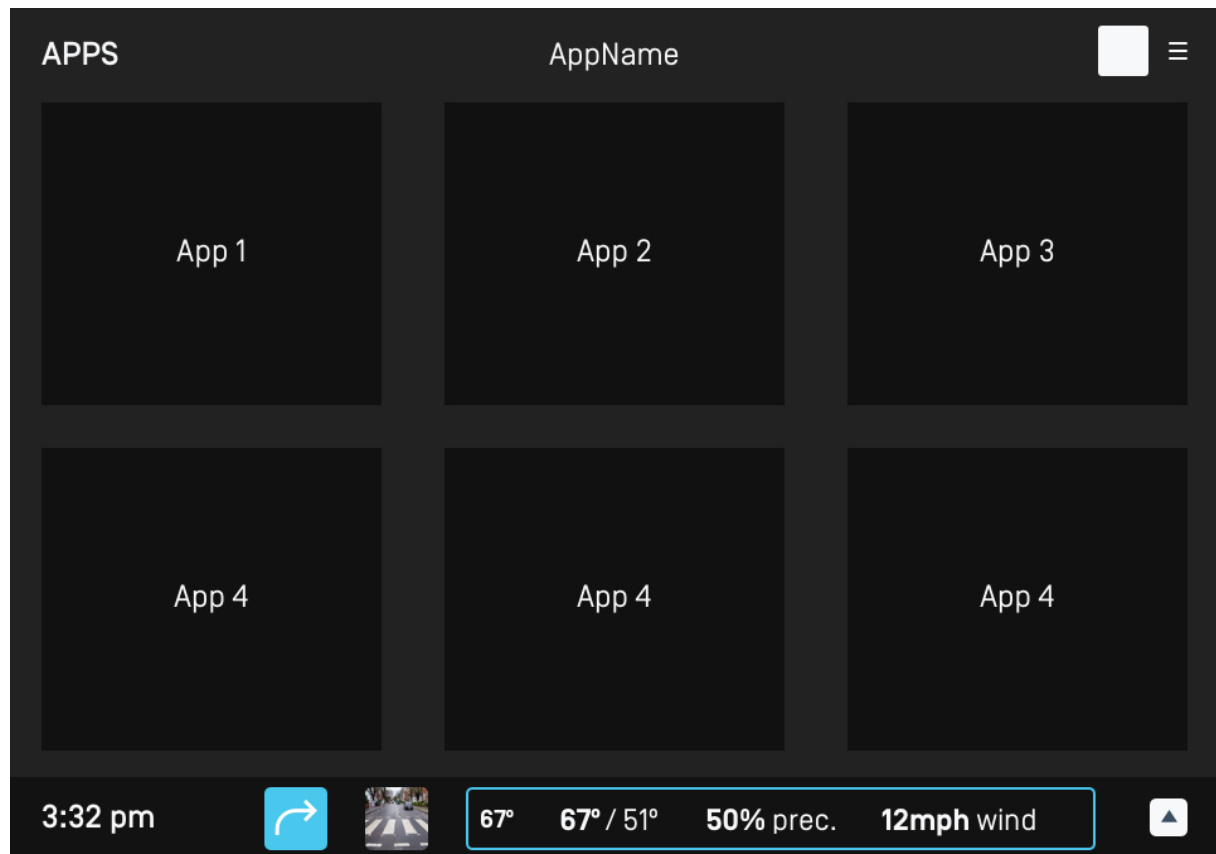
MEDIA EXPANDED

The expanded media view shows additional information using `mediaTitle` and `mediaArtist` while also providing basic playback controls that will send `OnButtonPress` RPCs to the app service provider.
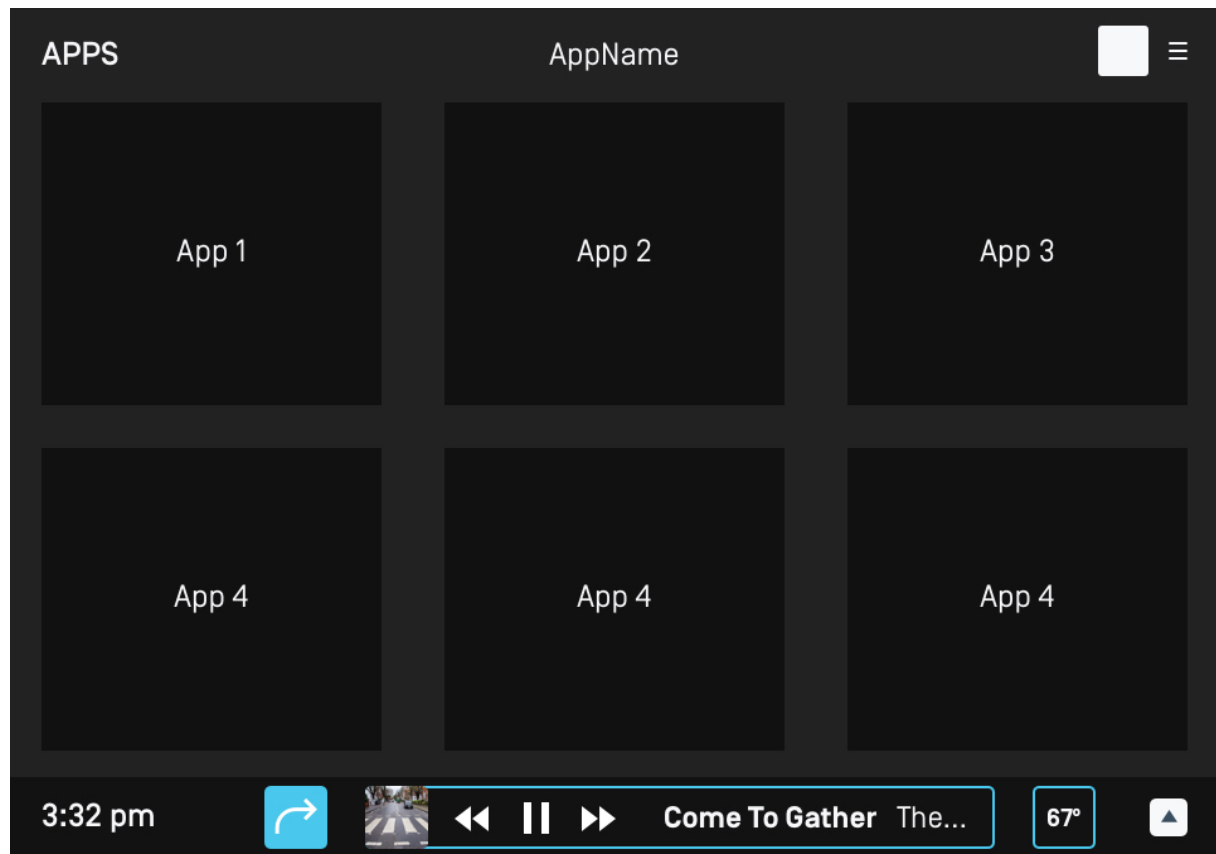
EXPLODED VIEW

The exploded view demonstrates how more information could be used from the app service providers to populate large UI elements.
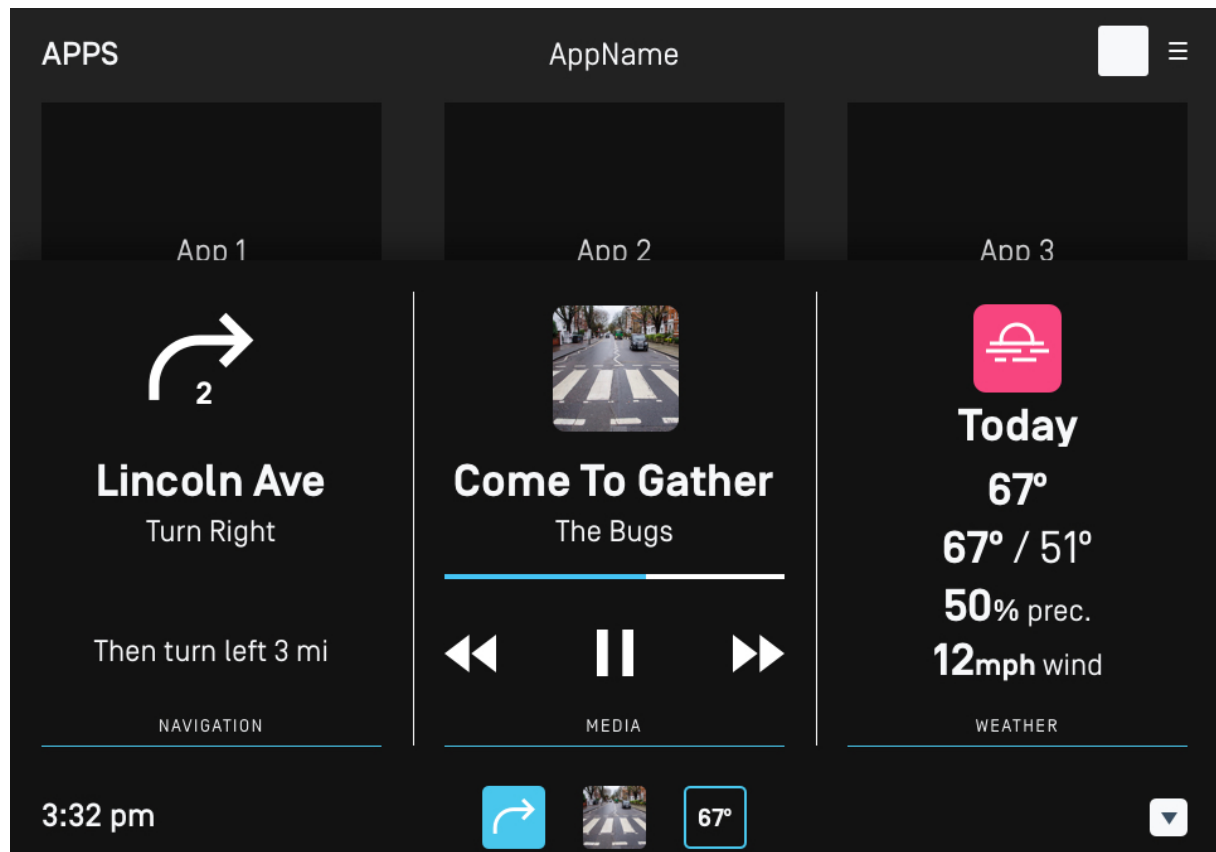
Protected Services

RELATED EVOLUTION PROPOSALS

- 0317: SDL Protocol Security Specification

1. Overview

Secured services in SDL are established using a TLS handshake, the TLS handshake process is defined by TLS and is not part of the SDL protocol.

SEQUENCE DIAGRAM

TLS Handshake Process
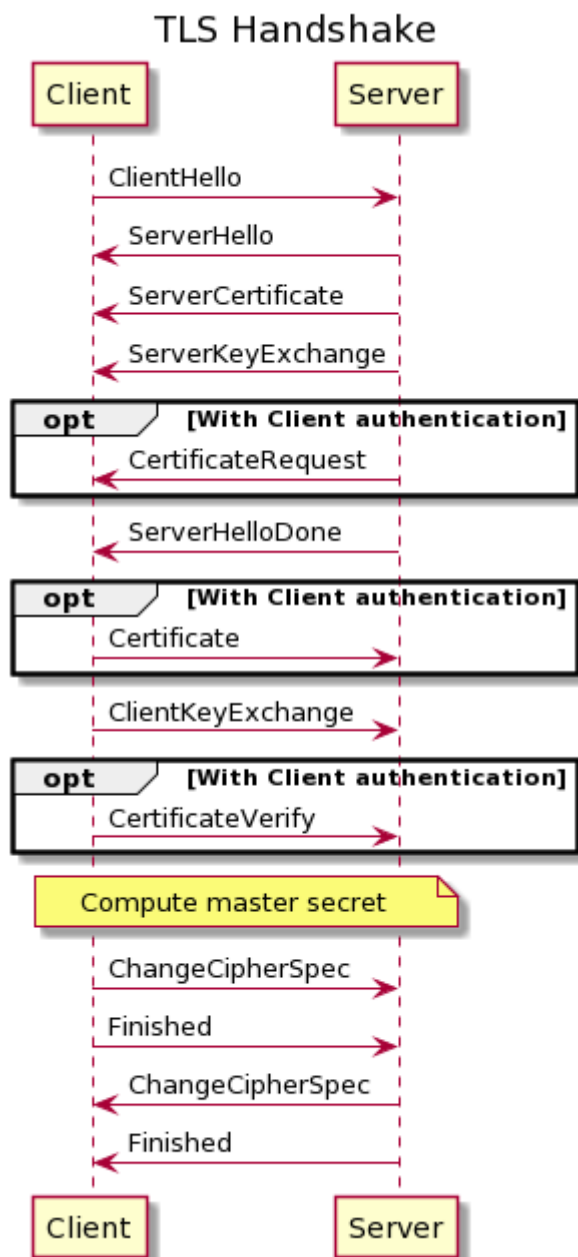
2. Peer roles during the handshake

The handshake for establishing secured services uses a client-server model which is configurable via SDL Core settings. An application must take the role of the server while SDL Core is the client. The client entity will initiate a TLS handshake with the corresponding security manager of the server. The client will do this only if the server was not previously authenticated in the current transport connection.

2.1 Peer certificate verification

According to the TLS handshake process, the peer certificate can be omitted for the server but is required for the client. Certificate peer verification can be enabled/disabled in SDL Core by changing the `VerifyPeer` parameter in the configuration file. The SDL app libraries do not require a certificate from Core for the TLS handshake, but Core does perform its own internal certificate validation before starting the handshake. During internal validation, Core checks if the certificate is missing (or outdated/invalid) and if so, it initiates a PTU to obtain a new certificate from the Policy Server. If a valid certificate can't be obtained, Core does not start the TLS handshake and it notifies the app library that starting the protected service has failed.

3. RPC processing

After RPC service encryption is enabled, SDL Core will reject any unencrypted RPC requests with an unencrypted response and result code `ENCRYPTION_NEEDED` if the RPC needs protection.

> **📝 NOTE**
>
> - SDL Core continues processing an unencrypted RPC request if the RPC does not need protection and responds with an unencrypted response.
> - SDL Core continues processing an encrypted RPC request if the RPC needs protection and responds with an encrypted response. In addition, SDL Core shall continue processing an encrypted RPC request if the RPC does not need protection and responds with an encrypted response.
> - SDL Core sends an unencrypted notification if the RPC does not need protection.
> - SDL Core sends an encrypted notification if the RPC needs protection.

RPC Encryption

RELATED EVOLUTION PROPOSALS

- 0207: RPC Message Protection

Overview

The RPC Message Protection feature is designed to encrypt specific RPC messages transmitted between a mobile application and SDL.

Determining When Encryption Is Needed

SDL Application

An SDL app is informed about the encryption requirement by an `OnPermissionsChange` notification sent by SDL Core. The `OnPermissionsChange` notification has a `requireEncryption` boolean parameter, which indicates the encryption requirement on the app level.

```
<function name="OnPermissionsChange" functionID="OnPermissionsChangeID"
messagetype="notification" since="2.0">
    <description>Provides update to app of which policy-table-enabled functions are
available</description>
    <param name="permissionItem" type="PermissionItem" minsize="0" maxsize="500"
array="true" mandatory="true">
        <description>Change in permissions for a given set of RPCs</description>
    </param>
    <param name="requireEncryption" type="Boolean" mandatory="false" since="6.0"/>
</function>
```

Each `PermissionItem` struct in the notification also contains a `requireEncryption` boolean parameter, which indicates the encryption requirement on the RPC level.

```
<struct name="PermissionItem" since="2.0">
    <param name="rpcName" type="String" maxlength="100" mandatory="true">
        <description>Name of the individual RPC in the policy table.</description>
    </param>
    <param name="hmiPermissions" type="HMIPermissions"  mandatory="true"/>
    <param name="parameterPermissions" type="ParameterPermissions"
mandatory="true"/>
    <param name="requireEncryption" type="Boolean" mandatory="false" since="6.0"/>
</struct>
```

Using the app level and RPC level encryption requirements, the app side will be able to determine whether the app needs encryption or not and which RPCs in particular will need

encryption.

An RPC message will require encryption/protection if the app has `requireEncryption=true` in the `OnPermissionsChange` notification, and the RPC has `requireEncryption=true` in the `PermissionItem` .

SDL Core

SDL Core uses the `encryption_required` flags in the policy table to determine whether or not an RPC message requires encryption.

The `encryption_required` flag, in the `functional_groupings` section of the policy table, indicates whether or not **all** the RPCs within a functional group require encryption.

JSON EXAMPLE

```
"RemoteControl": {
    "encryption_required" : true,
    "rpcs": {
        "GetInteriorVehicleData": {
            "hmi_levels": ["BACKGROUND", "FULL", "LIMITED"]
        },
        "SetInteriorVehicleData": {
            "hmi_levels": ["BACKGROUND", "FULL", "LIMITED"]
        },
        ...
    }
}
```

Whereas the `encryption_required` flag, in the `app_policies` section, indicates whether the app requires encryption or not.

JSON EXAMPLE

```
  "app_policies": {
    "default": {
      "encryption_required": false,
      "keep_context": false,
      "steal_focus": false,
      "priority": "NONE",
      "default_hmi": "NONE",
      "groups": [
        "Base-4"
      ]
    },
    "appid_12345": {
      "encryption_required": true,
      "keep_context": false,
      "steal_focus": true,
      "priority": "NONE",
      "default_hmi": "NONE",
      "groups": [
        "Base-4", "RemoteControl"
      ]
    },
    ...
}
```

SDL Core informs an SDL application about the encryption requirement via an `OnPermissionsChange` notification.

Handling Unencrypted Messages

SDL Core sends this result code to an SDL app when it receives an un-encrypted RPC request message **that needs encryption**.

```
<enum name="Result" internal_scope="base" since="1.0">
...
  <element name="ENCRYPTION_NEEDED" since="6.0">
    <description>SDL receives an un-encrypted RPC request that needs protection.
</description>
  </element>
</enum>
```

Setting Up Encryption

SDL Application

For more information on how to setup encryption on the SDL application side, please take a look at the Encryption guide for your selected platform.

- Android Encryption Guide
- iOS Encryption Guide
- JavaSE Encryption Guide
- JavaEE Encryption Guide

SDL Core

For more information on how to enable and utilize RPC Encryption within SDL Core, please take a look at the RPC Encryption Setup Guide section in the core guides.

Display Information

Display vs Voice

Designing your application on a mobile device is entirely a visual process. However, designing application for in-vehicle experience requires closer consideration of driver distraction rules to ensure the safety of your application user. Below listed are some of the best practices you will need to consider while designing your application for in-vehicle use so as to eliminate the need to understand the driver distraction laws and guidelines that are continually changing around the world.

General Guidelines

USE VOICE COMMANDS

It is highly recommended to focus on voice interactions first and foremost. In a vehicle, especially while driving, your application usage should be almost entirely accomplished using voice commands.

PRESENTING CLEAR INFORMATION

There are only a few ways of presenting information to the user. The first way is to write to the available *templates*, for example to your application's main screen. While more advanced displays allow for more updated fields and *templates*, you are always given a *base template* to write to.
The second is to send a text string for the voice engine to process and speak back to the user. The combination of the two, display *template* updates and sending text to be processed into speech, is the best way of presenting information to the user. An example of this is requesting a pop-up with text-to-speech (*TTS*), or ` Alert `.

You can find a list of available templates on our Supported Templates page.

USING DISPLAY

There are anywhere between two and four lines of text available to your application via the `Show` command. As a general rule, use this display to convey the current state of your application.

line 1 and *line 2*: Use *line 1* and *line 2* for relatively continuous updates such as, what station is playing, artist/track being streamed, distance to location, etc.

line 3 and *line 4*: Other information which might be useful, but not pertinent to functionality are best included on *Lines 3* and *4*, if available.

NUMBER OF CUSTOM SOFTBUTTONS

When using `Show` in combination with `SetDisplayLayout`, depending on the OEM, the number of custom softbuttons on each display layout will differ. The standard format of softbuttons for each system is as follows:

LAYOUT	STANDARD # OF SOFTBUTTONS
DOUBLE_GRAPHIC_WITH_SOFTBUTTONS	8
GRAPHIC_WITH_TEXT	0
GRAPHIC_WITH_TEXT_AND_SOFTBUTTONS	2
GRAPHIC_WITH_TEXTBUTTONS	3
GRAPHIC_WITH_TILES	3
LARGE_GRAPHIC_ONLY	0
LARGE_GRAPHIC_WITH_SOFTBUTTONS	8
MEDIA	8
NON_MEDIA	8
TEXT_AND_SOFTBUTTONS_WITH_GRAPHIC	2
TEXT_WITH_GRAPHIC	0
TEXTBUTTONS_ONLY	8
TEXTBUTTONS_WITH_GRAPHIC	3
TILES_ONLY	7
TILES_WITH_GRAPHIC	3
LAYOUT	STANDARD # OF SOFTBUTTONS

PROVIDE FEEDBACK

- Brand new user

  A welcome message with basic instructions could be given the first few times a user uses the application in vehicle using the Speak function. Once they've used the application a set number of times, you can remove these helpful prompts. Send a Show command to update the display with a welcome message or initialization message such as *"Buffering…"*.

- Logged In User Account

  If your application does require a logged in user account, you should always have logic to catch applications that are running in-vehicle without an active account and display a message notifying the user to log in when not driving.

  Refer to the Standard Interaction Phrases section for sample voice and/or text messages.

- Waiting for Online Data Download

  If your application is waiting for online data to get downloaded, display "*Loading...*" text on *Line 1* of the *template* to inform the user of the download.

- Background Application

  If your application is a background application, with no real action for the user to take while using it in-vehicle, you might not have any voice commands or soft-buttons available in your application. It is recommended to provide an "*About*" or "*Info*" Menu/voice command or/and soft-button (with voice command) which when used by the user provides a brief message about the application. You may also choose to point them to the help section in your mobile app which might be more descriptive.

- Dealing with Permissions

  If your application is using certain functions, such as vehicle data, it is essential to ensure that the application provides feedback to the user in cases when:

  - All or part of the vehicle data is not available in the vehicle (as some car lines or model only provide a subset of the vehicle data available in other car lines).
  - User did not provide consent to use vehicle data.
  - User has disabled access to all or part of the vehicle data your application is using.

  For the above scenarios, as a feedback you may let the user know that the application might not work as expected due to the lack of availability of vehicle data. Refer to the Standard Interaction Phrases section for sample voice/text messages.

Note that some RPCs are protected by policies for every OEM. To gain access to such RPCs, contact the OEM.

- Using ScrollableMessage

  `ScrollableMessage` is a useful way to show and display text to a user. In some markets like Europe and Asia, `ScrollableMessage` can be used at any time, to show a message or other relevant information that wouldn't fit on an `Alert`. It is important to note that in North America, `ScrollableMessage` is limited in use and cannot be used while the vehicle is in motion. Please be advised that you may find these messages are blocked from being shown in that region. To detect this condition, use the `OnDriverDistraction` notification on vehicles before showing the message, or if you detect `ScrollableMessage` is rejected due to driver distraction restrictions, simply use a `Speak` or another method to convey the information.

- Handling API rejections Part of guides

  Depending on the state of the system, or your application permissions, some of the messages you send to the vehicle may get rejected. It is important to understand why these are happening when they occur, so that you may present the proper information to the user. If a command you send is `REJECTED`, it is recommended to send it again, depending on the message. For example, if an `AddCommand` is rejected, you will need to send it again, to ensure your menu option or voice command is available on the system. If a request for `Alert` is `REJECTED`, it may mean the system is in a state in which your message cannot be played. You may want to try again after 15-30 seconds if the information is pertinent to the user.

- General feedback recommendation

  - Applications must provide an audio or visual response to user interactions (button presses, *VR*, `AddCommands`, etc.).
  - Applications must not provide an incorrect or unexpected response to user interaction.

- Standard Interaction Phrases

  - Login Required
  - Text: To use < *app name* > you have to be logged in.
  - Voice: To continue using < *app name* > please login on your mobile phone while not driving.
  - Vehicle Data Availability
  - Text: Grant access to vehicle data in Mobile Apps settings.
  - Voice: < *App name* > might not work as expected as we are unable to access < *Vehicle Information, Push Notifications, Location Information and Driving*

Characteristics >. Please enable this feature in Mobile App's settings menu while not driving.

Graphics

When implementing graphics into your apps you should make sure that the app looks good in both day and night mode. For graphics this implies that you should at either have a background or work with outlines. For icons, outlines are needed to increase contrast in all situations. Light icons should have a dark contrast and dark icons should have a light contrast.

Please ensure to use the correct dimensions for your images.

Graphics used in `choiceSets` must be uploaded before the `choiceSet` is used. This will decrease performance if many different graphics are to be used in one `choiceSet`. To mitigate that the usage of generic icons is encouraged (like a CD icon instead of the actual Album art).

More information on using images with SDL is available in the Uploading Images guide available for either iOS or Android.

Driver Distraction Rules for Graphics

Due to driver distraction rules the graphics used can't incorporate text or any form of data or graph. No moving images or video can be used, or any graphic that a user could interact with. If your app incorporates social media you are not allowed to display any graphics from social media posts or any form of attachment.

Languages

General Guidelines

As a developer, you will want to localize your application, via voice and on the display to match the experience with the rest of the user interface inside of the vehicle.

LOCALIZATION AND HANDLING LANGUAGE CHANGES

In order to properly handle language changes inside of the vehicle, we propose the following rules.

1. Register your application with the language it currently supports on the phone's user interface.

2. Check the display and voice language that the vehicle is currently set to in the `RegisterAppInterface` response.

3. If your application does not support the language in the vehicle, you may send commands as usual. By default if the user starts your application in the vehicle, the vehicle will notify the customer the languages between the vehicle and phone does not match and no other steps are required.

4. If your application supports the language(s) in the response value, send a `ChangeRegistration` message with the vehicle language and proceed to adding your commands and bootstrapping your application.

DYNAMIC SWITCHING OF LANGUAGES:

The best practice for your SDL integration is a dynamic switching of the Apps SDL language. The `RegisterAppInterface` RPC will report the head unit's language. If this language differs from the current language your App is set to, you should dynamically reload the strings in the language to that set in the head unit. The driver is used to the language in his car, therefore your app should be aligned with the language in the vehicle.

Note: If dynamic switching of languages is not a possibility for your application (because you load strings dynamically from your backend based on the user profile language settings), you may still choose to keep your language. The user will be informed from the head unit that the App's language differs from SYNC and that Voice commands will not wo

rk as expected. Check out Regional Language Switching section for best practices to minimize false notifications to the user.

APP'S TTSNAME

It is always important to keep in mind the voice engine in the head unit will always pronounce the given data according to the language it is set to.

```
Example:
App Name "Livio Music" sounds like "Leeveeo Moozik" (which is not how the app name
is supposed to be pronounced) if the head unit is set to German.
```

To prevent your app name from being pronounced in from happening and for the user to be able to correctly activate your app via voice as well as for the head unit to correctly pronounce the App Name, you will have to use the two parameters `TTSName` and `VRSynonyms` in the `RegisterAppInterface` . Both fields have to be filled with a `String` that represents the pronunciation of your App name in the current language.

```
Example:
Fill `TTSName`and one value of `VRSynonyms` with `Leeveeo Moozik` when the
Livio Music connects to a German head unit to get the correct pronunciation
of the app name in German. This is the only way to enable the user to start
the app with the english pronunciation of Livio Music and also allow the head
unit to pronounce name of the app correctly.
```

Note: The recommended best practice mentioned above must be followed only if the app name sounds wrong in the language the app switches to.

REGIONAL LANGUAGE SWITCHING

If your app only supports one variant of English:
You should update the registration information using the `changeRegistration` RPC (you don't have to re-register, because you don't have to change the `TTSName`). As soon as you get the information that the user's vehicle is set to a different regional version of the language, you should call the `ChangeRegistration` RPC with the actual regional variant that the user's vehicle is set to. There is no other change required.

```
Example:
App is set to "EN_US".
RegisterAppInterface reports the head unit is set to "EN_GB".
Use "ChangeRegistration(EN_GB, EN_GB)" to update the language.
Note: You don't have to update either "TTSName" or "VRSynonyms".
```

PERSISTED LANGUAGES

If your App's languages differs from your SDL languages, the App should store this different language and use it the next time it registers with the head unit.

The above recommendation also covers the use case where the user has his vehicle always set to a different language than the language on his mobile phone. If the app did not persist the language it would have to switch every time, which degrades the apps performance.

Menu Items and Interactions

All the major features or functions of your application (nouns/verbs) could be loaded as top-level voice commands and or menu items using `AddCommands`. It is recommended that this list be less than 150 commands, to improve application initialization performance and for high voice recognition quality.

General Guidelines

RESPONDING TO USER INPUT

As an application, you will want to respond to user input. To limit changes in modality, we highly recommend looking at the trigger source for input provided to your application, either via voice or menu. In instances where `OnCommand` indicates you've made a selection, and you want to use `PerformInteraction`, ensure the mode is the ***same***. For example, if trigger source is `MENU`, you'll want to start your interaction with an interaction mode of `MANUAL_ONLY`. If trigger source is `VR`, you'll want to trigger your interaction as `BOTH` or `VOICE_ONLY`. Also, as a best practice always set your `Perform Interaction` timeout to max giving sufficient time for the user to respond.

HANDLING CHOICES, CHOICESETS AND COMMANDS

Applications can load `ChoiceSet` with 100 items. If your `PerformInteraction` requires additional commands, you may reference additional `ChoiceSets`. That is, one `PerformInteraction` can contain more than one `ChoiceSet`. If your list of choices is known ahead of time, it is helpful to create these during your initialization phases, and simply reuse the `ChoiceSet` throughout your application's lifecycle. When a user initiates an interaction, the user may choose from whatever choices are defined at that moment. It is up to the application to ensure that all appropriate choices are defined ***before*** the app interaction. Also, consider grouping your choices in a way that maximizes reusability of the defined `Choices` or `ChoiceSets`.

ORDERED ITEMS

Always order items in your lists for better user experience. Ordering the items which are more often used in your application or ordering the items based on the specific user preferences, or based on current location of the user etc. will help keep the most important items to the user accessible. In general, order the items from most important to least important.

VR FOR SOFTBUTTONS

To add VR for softbuttons, use `AddCommand` with no `menuParameters`.

NAME AND IMAGE

You may use both name and image for items wherever applicable.

CHOOSING VOICE GRAMMAR

Applications must not register voice grammars using synonyms that include other application names, or conflict with on-board voice commands.

MAKING LISTS MORE INFORMATIVE

It is always a good practice to add relevant and useful information to the list items, for example if your app is a media app and you have a list of audio contents, adding information such as when the content was aired.

HANDLING LISTS

PerformInteraction lists should always be as small as possible by UX design. However, in cases where having long lists cannot be avoided, please follow the below best practices for better user experience.

Note: Avoid opening `PerformInteraction` from within another `PerformInteraction` .

1. List with multiple action item per choice

   - For each item which has several more actions available:
     Example: Searching for events in the vicinity. There might be a big number of events. On each event you can select "Call venue", "Navigate to it", "Details", "Play" (playing the artist of a potential concert in the vicinity).

     - Present each result on its own screen via `Show` and announce it with a pure voice alert via `Speak`.

   - Define softbuttons for each possible action and add the respective command to voice and menu.
   - To cycle through the result list, you may use one of the below options:
   - `Skip` button (which is currently only available for Media Apps)
   - Softbuttons & voice commands
   - Show the current item and the length of the list in the Media-track. For example 1/10 for the first item out of 10 items from the result.

- Announce the availability of Skip hard button on the steering wheel for easy navigation.

Note: Only make the announcement the first time the search has been completed to reduce any kind of annoyance to the user.

- For each item only one action available:
  Example: scenario is already "Navigate To _"
  In contrast to the above, you may choose to use PerformInteraction.

2. Simple lists
   If the results are known by the user or the result list is very small you can use PerformInteraction instead depending on the current use case.

Example:

```
User Action 1:

VR/MENU: "Create result list"
Result 1:

 SCREEN LINE 1:  "Result 1 a"
 SCREEN LINE 2:  "Result 1 b"
 MEDIA TRACK:    "1/25"
 SOFTBUTTONS:   Action 1, Action 2, Action 3,...,Previous, Next, More...
 TTS:          "Result 1 <possibly more information about the result>"
            //First Time: Announce possible ways to navigate (skip buttons,
              voice command, etc..)
```

User Action 2:

VR/MENU/SOFTBUTTON: "Action 1"
Result 2:

This is only applicable if the "Action 1" of "Result 1" results in another
screen. If just a call is initiated by "action 1" there is no need to
display an ensuing screen.

  SCREEN LINE 1: "Action 1 Result 1 a"
  SCREEN LINE 2: "Action 1 Result 1 b"

  SOFTBUTTONS: Softbuttons highly depend on the use case you are developing.
            If the user again has an array of choices, use the same
            concept as explained above. Otherwise, adapt the screen to
            your use case.
  TTS: "Result of Action 1 on Result 1"

User Action 3:

VR/MENU/SOFTBUTTONS: "Next"
Result 3:

  SCREEN LINE 1: "Result 2 a"
  SCREEN LINE 2: "Result 2 b"
  MEDIA TRACK: "2/25"
  SOFTBUTTONSS: Action 1, Action 2, Action 3,...,Previous, Next, More...
  TTS: "Result 2 <possibly more information about the result>."

PerformAudioPassThru

The `PerformAudioPassThru` RPC feeds you audio data from the vehicle's microphone.
The audio data can be used in cloud-based and on-line voice recognition to achieve
dynamic user interaction, such as POI (point of interest) search, information query, or even
record when the driver is singing. The audio data will be in uncompressed PCM format.
The sampling rate, bit width, and timeout can be set, however, the supported parameters

will be sent in the `registerAppInterface` response. Generally, 16 bit width, 16kHz sample rate will be supported.

The parameter `muteAudio` is used to define whether or not to mute current audio source during AudioPassThru session.

When the `PerformAudioPassThru` is used for voice recognition, `muteAudio` should be set to true to minimize audio interference.

If you want to mix the input audio from `PerformAudioPassThru` session with current audio source, eg. a karaoke app recording both the user's voice and the background music, you can set `muteAudio` to false.

`onAudioPassThru` keeps you updated with the audio data transfer every 250ms.

`EndAudioPassThru` enables you to end the audio capture prematurely. This is useful if your app analyzes the audio level and detects that the user has stopped speaking.

Additional notes about the audio data format:

- There is no header (such as a RIFF header).
- The audio sample is in linear PCM format.
- The audio data includes only one channel (i.e. monaural).
- For an 8 bit width, the audio data is unsigned. For a 16 bit width, it will be signed and little endian.

Audiobooks

An audiobook app for SDL should include as much of the native apps functionality as possible, since that is what a user will be expecting out of the app.

USE VOICE COMMANDS

Your Audiobook app should contain such basic functionality as `Play` , `Pause` , and `Resume` . You can use build in controls for these features by calling the `subscribeButton` RPC. This will allow you to not only create buttons on the infotainment system but also be informed when the user presses hard buttons, which might be placed on the steering wheel. It is important to also add voice control capabilities for these functions. To add a voice command use the `AddCommand` RPC but omit the `menuParams` argument. This will create a voice command without creating an item in the menu.

More information about this RPC can be found on the UI.AddCommand page.

DISPLAY INFORMATION

Your app should display information like title, author, and potentially additional items like chapter number. It is encouraged to use the following layout of information on the screen:

TEXTFIELD	INFORMATION
mainField1	Title
mainField2	Author
mainField3	Chapter (x/x)

It is encouraged to show static images of the book cover, however moving or interactive images are not allowed. Neither is the display of the actual text of the book.

More information can be found in the DisplayInformation section.

MANAGING LISTS

Use the `PerformInteraction` RPC to allow users to choose from lists of Books.

MANAGING AUDIO

When you receive the `NOT_AUDIBLE` state you should pause the audio and resume when you receive `AUDIBLE` . The pausing and resuming should be independent on the current `HMILevel` state.

Music Apps

When creating a music app for SDL the app should include as much of the native apps functionality as possible. If your app requires a user name and password to be used then the user should be informed for this instead of the user just being unable to access the apps functionality. Please refer to the `Displaying Information` section.

BASIC COMMANDS

Your Music app should contain such basic functionality as `Play` , `Pause` , and `Resume` . You can use build in controls for these features by calling the `subscribeButton` RPC. This will allow you to not only create buttons on the infotainment system but also be informed when the user presses hard buttons, which might be placed on the steering wheel. It is important to also add voice control capabilities for these functions. To add a voice command use the `addCommand` RPC but omit the `menuParams` argument. This will create a voice command without creating an item in the menu.

DISPLAY INFORMATION

Your app should display things like song name, artist name, and optionally the album name. It is encouraged to use the following layout of information on the screen:

TEXTFIELD	INFORMATION
mainField1	Song
mainField2	Artist
mainField3	Album

It is encouraged to show static images of the album art, however moving or interactive images are not allowed neither is the display of lyrics.

Below is an example of the Livio Music Player home screen using the MainFields and displaying a graphic.
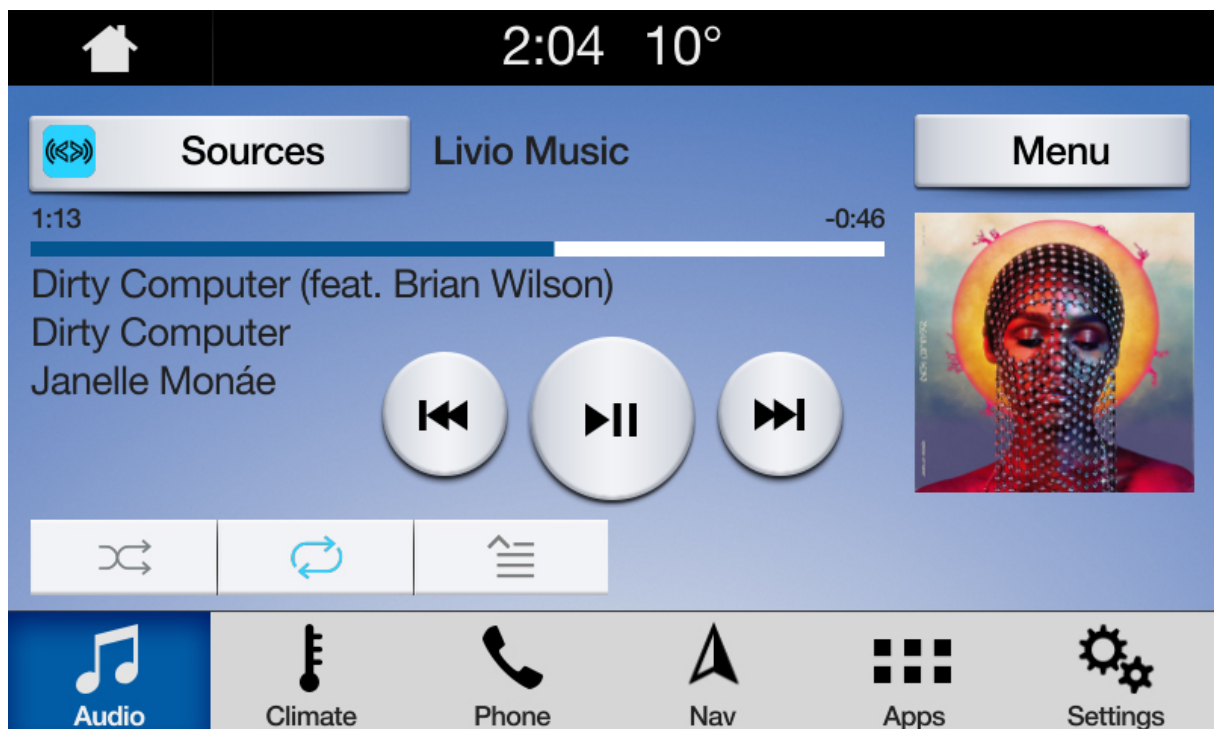
USING PRESETS

If your app features different stations or streams you should consider allowing the user to save these stations to preset buttons. To receive notifications about preset button presses please use the `subscribeButton` RPC with the respective presets as arguments. On some SDL implementations you will be able to change the text on a preset button. To change this text use the `customPresets` array in the `show` RPC.

MANAGING AUDIO

When you receive the `NOT_AUDIBLE` state you should pause the audio and resume when you receive `AUDIBLE`. The pausing and resuming should be independent on the current `HMILevel` state.

Application Data Resumption

Data resumption is used when an application requests to restore data used in the previous ignition cycle after an unexpected disconnect. Resumption starts during the `RegisterApplnterface` request message processing. The application must include the hashID parameter in the request for SDL Core to recognize a previous connection and attempt to restore the previous HMI state. If the hashID received from mobile is correct, SDL Core will restore the following data to the HMI.

- Menu and VR Commands
- Sub Menus
- Choice Sets
- Persistent Files and Images
- Global Properties
- Button Subscriptions
- App Service Subscriptions
- Vehicle Data Subscriptions
- Interior Vehicle Data Subscriptions
- Remote Control Module Consent

For data resumption purposes, SDL Core must store application-related data such as menu commands, application global properties, voice recognition data, and subscriptions

for previous ignition cycles after an unexpected disconnect or ignition off.

> **🖊 NOTE**
>
> On every fourth ignition cycle, SDL Core clears all corresponding application-related data used for resumption. This will either be stored in app_info.dat or in a resumption database depending on SDL Core's ini configuration.
>
> smartDeviceLink.ini example
>
> ```
> ; Resumption ctrl uses JSON if UseDBForResumption=false for store
> data otherwise uses DB
> UseDBForResumption = false
> ```

HMI must store the VR grammars compiled for applications that are unregistered by an unexpected disconnect or ignition off. During data resumption, the HMI may also have to resume the previous audio source. Refer to `BasicCommunication.OnResumeAudioSource` .

If the application resumes data successfully:

- SDL Core will reply to the mobile application's `RegisterAppInterface` request with `resultCode = SUCCESS`
- SDL Core will provide the HMI `BasicCommunication.OnAppRegistered` with `resumeVrGrammars = true` to notify the HMI that `VRGrammars` must be resumed. On this event, the HMI must restore the application related VR grammars for the appID received via an `OnAppRegistered` notification.
- SDL Core must restore application-related data and send to the HMI after an `OnAppRegistered` notification.
- Applications will not need to resend the following RPCs:
- AddCommand (Menu + VR)
- AddSubMenu

- CreateInteractionChoiceSet
- CreateWindow
- SetGlobalProperties
- SubscribeButton
- SubscribeVehicleData
- SubscribeWayPoints
- PutFile
- GetInteriorVehicleData (if previously sent with `subscribe = true`)
- GetSystemCapability (if previously sent with `subscribe = true`)
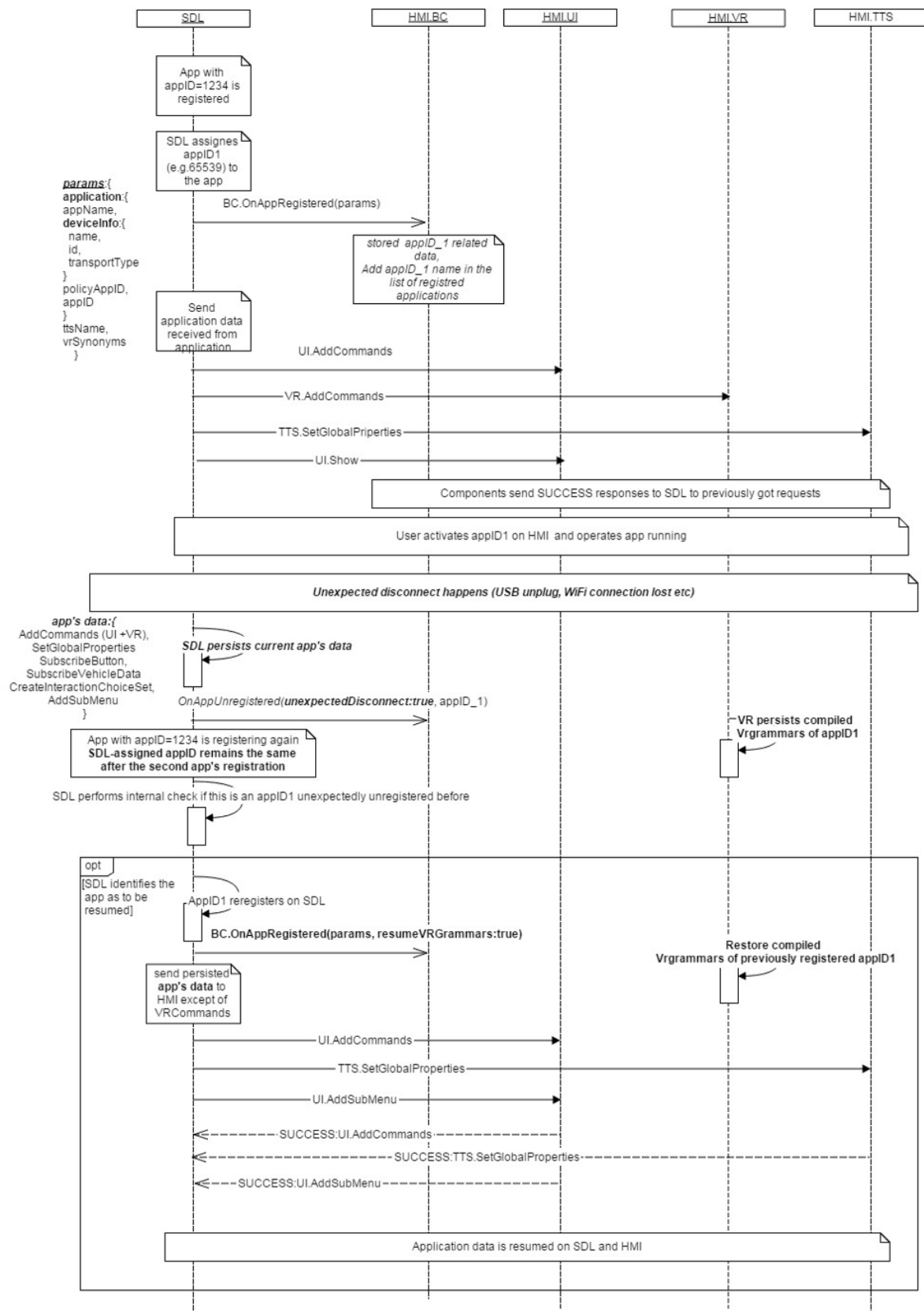- GetAppServiceData (if previously sent with `subscribe = true`)

SEQUENCE DIAGRAM

Successful App Resumption

View Diagram

If the HMI responds with an error to any of the resumption requests for an application sent by SDL Core, SDL Core will revert any data that has been resumed for the application thus far (ex. by sending DeleteCommand message for all successful AddCommand

requests). Any such error response from the HMI indicates an unsuccessful resumption, and SDL Core will continue to register the app in a fresh state.

If the application does NOT resume data successfully:

- SDL Core will reply to the mobile application's `RegisterAppInterface` request with `resultCode = RESUME_FAILED`
- SDL Core will provide `OnAppRegistered` with `resumeVrGrammars = false` or no resume parameter at all.
- SDL Core cleans up all previously stored application data for the application that failed to resume. The HMI must also clean up previously compiled `VRGrammars` for the application.
- The application will send new data to start SDL operations. In this event, SDL Core and the HMI should restart the cycle of collecting application data for resumption.
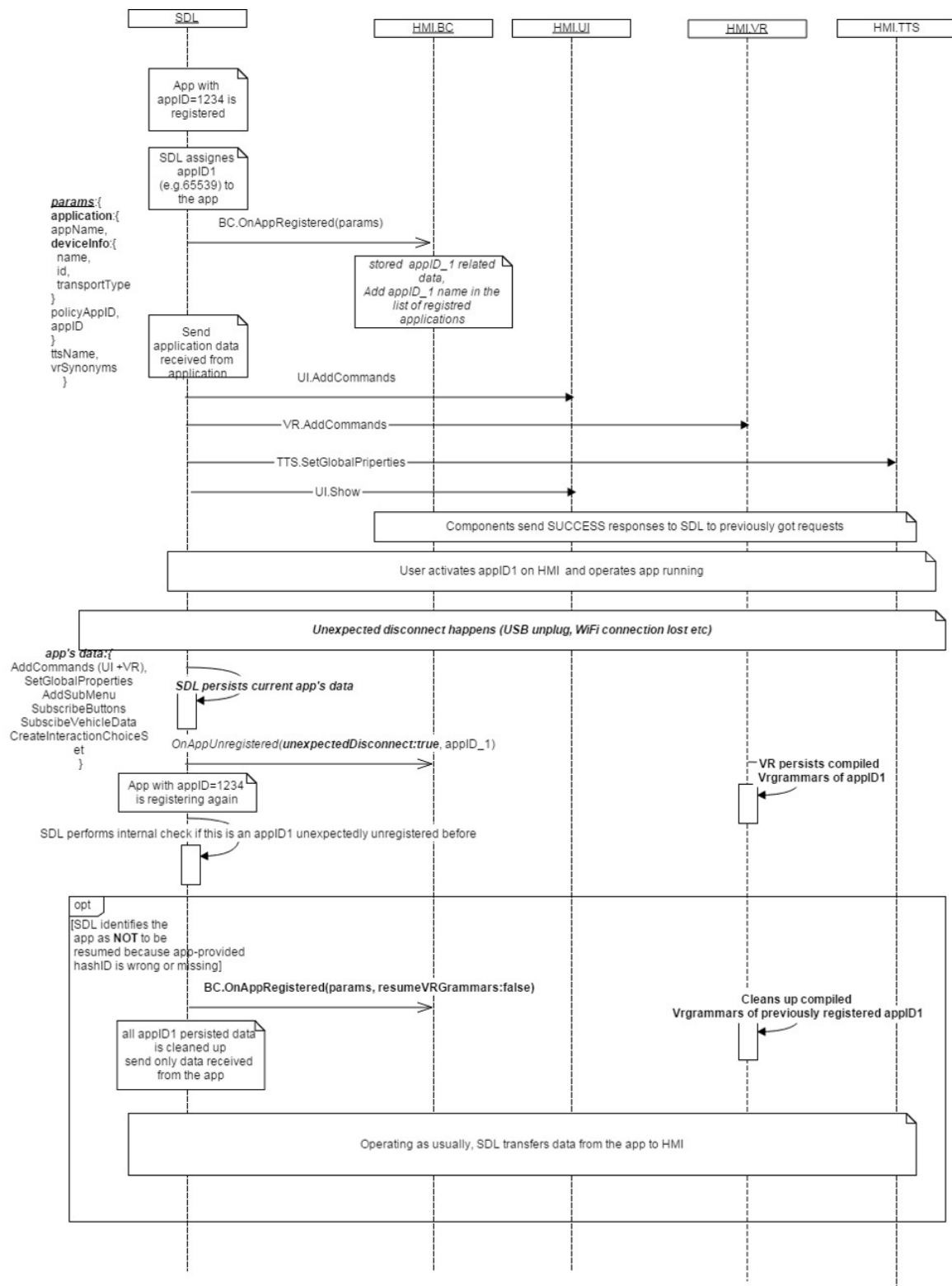
SEQUENCE DIAGRAM

Unsuccessful App Resumption

View Diagram

SmartDeviceLink FAQ

Here are a few of the most common questions new developers have around the SmartDeviceLink project.

- Is WiFi a supported transport?
- Can I implement custom HMI templates?
- Can I implement custom vehicle data messages?
- What is the process for obtaining an App ID?
- Why does SDL require that I use templates for non-navigation applications?
- I didn't find the answers I was looking for, where else can I look?

Is WiFi a supported transport?

WiFi transports are supported in two ways with various limitations. First, the production version is only available as a secondary transport. This means that a primary transport (such as Bluetooth or USB) has to be connected first in order for the TCP transport to connect.

Secondly, this transport can be used in the development and debugging phase of SDL integration and will act as the primary transport in that situation. For more information please see the evolution proposal SDL 0141: Supporting simultaneous multiple transports.

Can I implement custom HMI templates?

This is possible, but not recommended as any app that builds itself for the head unit's custom template would not work with any other systems. If new templates are desired, they should go through the SDL Evolution Process and be adopted by the project.

Can I implement custom vehicle data messages?

This is possible, but not recommended as any app that builds itself for the head unit's custom vehicle data types would not work with any other systems. If new vehicle data types are desired, they should go through the SDL Evolution Process and be adopted by the project.

What is the process for obtaining an App ID?

To obtain an App ID, you must first register for an account on the SDL Developer Portal, and then register the company to which the app should belong. Once your company is registered, you can select the App IDs tab from your company profile and click the "Create New App ID" button to provide your app information and obtain your App ID.

Why does SDL require that I use templates for non-navigation applications?

Templates are the best way for you to design your application with the driver's safety in mind. For more information on the benefits of templates, please see this document.

I didn't find the answers I was looking for, where else can I look?

Each project has documentation, guides, and may have their own FAQ that can help. If you still need help, please join the SmartDeviceLink Slack team and ask your question.