sdl

HMI Documentation

Document current as of 05/24/2019 12:01 PM.

Overview

This document provides the information for integrating the SmartDeviceLink (SDL) Embedded Component (generally referred to as SDL Core) with vehicle Head Unit (HU)

SDL is a system that is installed on the HU and allows a connected application on the mobile device to utilize some vehicle functionality such as text-to-speech, voice recognition, display, hardware buttons, etc.

With SDL as the connectivity middleware in a vehicle's infotainment system, applications running on connected iOS and Android devices might be presented to the vehicle HMI.

This guideline describes:

- The types of transports supported by SDL for communication with the HMI
- How to connect over one of the supported transports
- The types of messages and message formats used for communication
- The API that needs to be supported by the vehicle HMI
- Drawings showing exemplary display layouts for illustrating expected HMI behavior
- Sequence Diagrams showing the sequence of messages to be sent and received between SDL and the HMI
- Examples for each function call in different message formats corresponding to the transports currently supported

How to use this guide

The SDL HMI Integration guidelines may seem daunting at first, but we're not asking you to read them from cover to cover at the start of your exploration into SDL. We recommend an implementation of SDL HMI as follows

1. Get connected to SDL by following the Getting Started portion of these guidelines
2. Once you're connected and registered, connect a sample app such as the iOS RPC Builder to start to understand the RPC messaging layer in SDL between core and your HMI.
3. From there, you'll notice messages such as BasicCommunication.UpdateAppList. Use this guide to understand what those messages mean and how they can be leveraged in your HMI to provide the best possible user experience.

Abbreviations and Definitions

Abbreviations used in this document are collected in the table below

ABBREVIATION	MEANING
ASC	App Service Consumer
ASP	App Service Provider
BT	Bluetooth
CID	Center Information Display
DID	Data Identifier
DTC	Diagnostic Trouble Code
ECU	Electronic Control Unit
GPS	Global Positioning System
GUI	Graphical User Interface
HDOP	Horizontal Dilution of Precision
HMI	Human Machine Interface
IVI	In Vehicle Infotainment
JSON	JavaScript Object Notation
PDOP	Positional Dilution of Position
RPC	Remote Procedure Call
SDE	Software Development Environment
SDL	SmartDeviceLink
SEE	Software Engineering Environment

ABBREVIATION	MEANING
TTS	Text To Speech
UTC	Universal Time Coordinate
VDOP	Vertical Dilution of Precision
VR	Voice Recognition

Getting Started

Component Readiness Confirmation

The HMI must register each component which can communicate with SDL using the following format

```
var JSONMessage = {
        "jsonrpc": "2.0",
        "id": -1,
        "method": "MB.registerComponent",
        "params": {
           "componentName": "UI"
        }
     };
```

The possible componentNames are `UI`, `Buttons`, `BasicCommunication`, `VR`, `TTS`, `Navigation`, and `VehicleInfo`

Once the components are registered, the HMI must notify sdl that it is ready to begin further communication using the BasicCommunication.OnReady notification.

Upon receipt of the OnReady notification, SDL will begin checking the availablility of the different HMI components via a chain a requests:

- `UI.IsReady` - The display availability
- `VR.IsReady` - The voice recognition module availability
- `TTS.IsReady` - The Text-To-Speech module availability
- `Navigation.IsReady` - Navigation engine availability
- `VehicleInfo.IsReady` - Indicates whether vehicle information can be collected and provided.

Registering for Notifications

The HMI must also register for notifications individually using the following RPC format

```
var obj = {
        "jsonrpc": "2.0",
        "id": -1,
        "method": "MB.subscribeTo",
        "params": {
           "propertyName": notificationName
        }
     }
```

Where propertyName is the name of the notification, such as `Buttons.OnButtonSubscription`

MUST

The HMI must register its components, send the OnReady notification, respond to each of the IsReady RPCs, and register for the notifications it would like to receive.

NOTE

If the response to any of the component `IsReady` requests contains `{"available": false}`, SDL will no longer communicate with that component.

IsReady Sequence Diagram

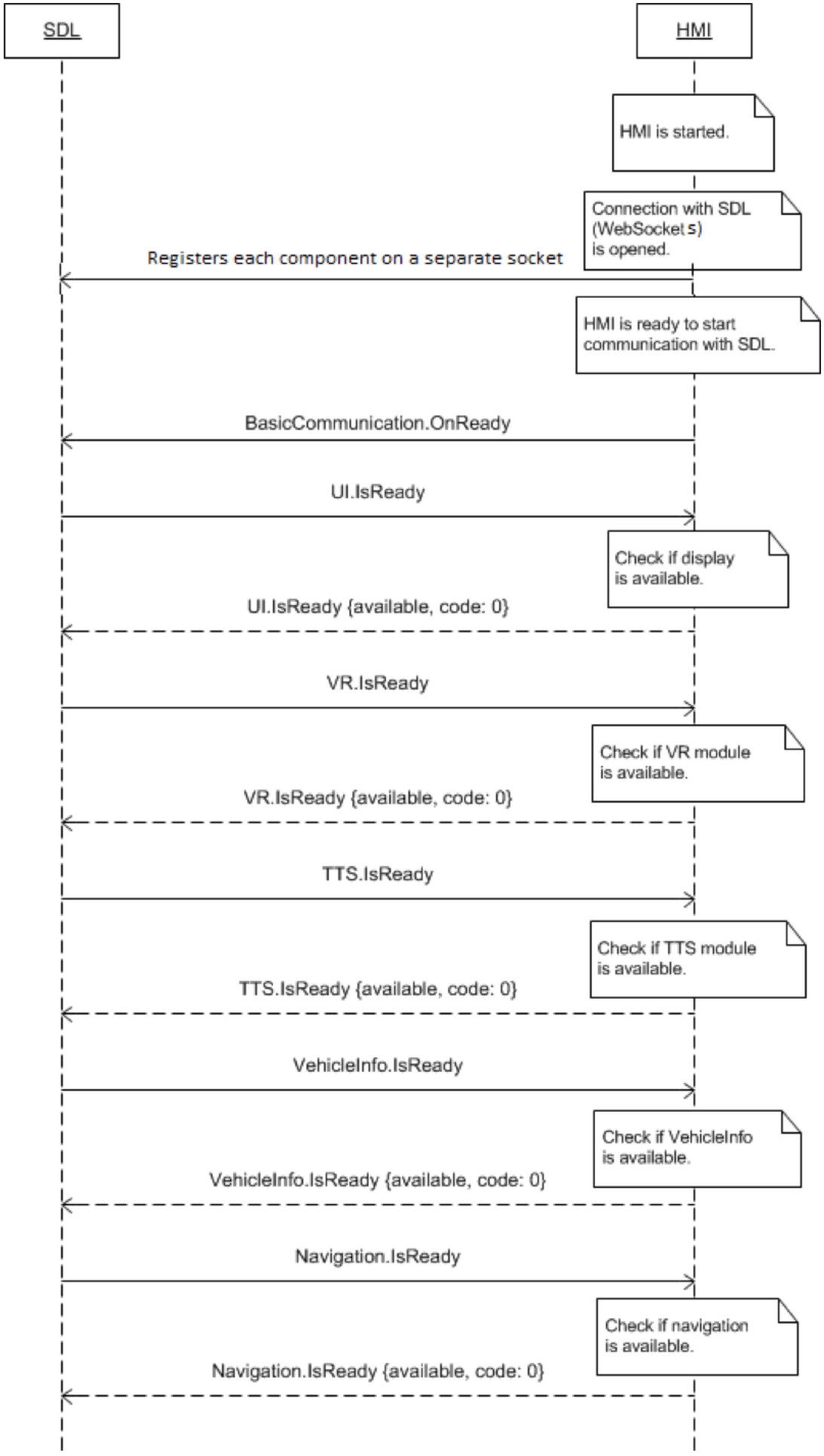
NOTE

In the case of a WebSocket connection, the RPCs to each of the components is sent within each separate WebSocket connection.

SEQUENCE DIAGRAM

IsReady Sequence

View Diagram


SDL
HMI
HMI is started.
Connection with SDL (WebSocket s) is opened.
Registers each component on a separate socket
HMI is ready to start communication with SDL.
BasicCommunication.OnReady
UI.IsReady
Check if display is available.
UI.IsReady {available, code: 0}
VR.IsReady
Check if VR module is available.
VR.IsReady {available, code: 0}
TTS.IsReady
Check if TTS module is available.
TTS.IsReady {available, code: 0}
VehicleInfo.IsReady
Check if VehicleInfo is available.
VehicleInfo.IsReady {available, code: 0}
Navigation.IsReady
Check if navigation is available.
Navigation.IsReady {available, code: 0}

Connecting to SDL

Requirements to HMI Adapter

WebSocket is the primary means of communicating with the SDL component from the vehicle. In a basic example, an HTML5 HMI would use a native WebSocket library to communicate with SDL Core.

The HMI Adapter must:

MUST

- be installed on the same vehicle HU OS where SDL is installed, or the HMI must be able to be networked to SDL and address it via a static IP address.
- create and initialize components which are defined in the HMI_API specification for the version of SDL which is running on the vehicle HU. (BasicCommunication, UI, Buttons, VR, TTS, Navigation, VehicleInfo)
- Establish a separate WebSocket connection with SDL for each of components defined in the HMI_API specification.
- Use the appropriate corresponding connection when sending responses and notifications to any connected component.

Handshake

For opening a WebSocket connection, a handshake must be performed.

NOTE

1. Client/Server relationship
    - SDL is the Server
    - The HMI is the Client
2. Host
    - SDL is listening on 127.0.0.1:8087
3. WebSocket Protocol Version 13 is used by SDL

HMI Component Registration

Once all the requested connections are opened, the HMI must send the JSON request for registering each component TODO: (see section 4.2 for JSON format details and Example #4 for registration example).

Component Registration has the following unique request requirements

KEY	VALUE INFO
"id"	a multiple of 100 (100, 200, 300, ..., 700)
"jsonrpc"	"2.0" - constant for all messages between SDL and the HMI
"method"	"MB.registerComponent" - the request is assigned to SDL's MessageBroker where the component name will be associated with the socket ID. Further, SDL will send messages related to the named component over the corresponding connection
"componentName"	The name of the component being registered. Must correspond to the appropriate component name described in the current guidelines.

SDL Provides a JSON Response

KEY	VALUE INFO
"id"	The value from the corresponding request
"result"	Value of id multiplied by 10. HMI can treat this as a successful registration

Component Registration Examples

REGISTER COMPONENT JSON MESSAGES

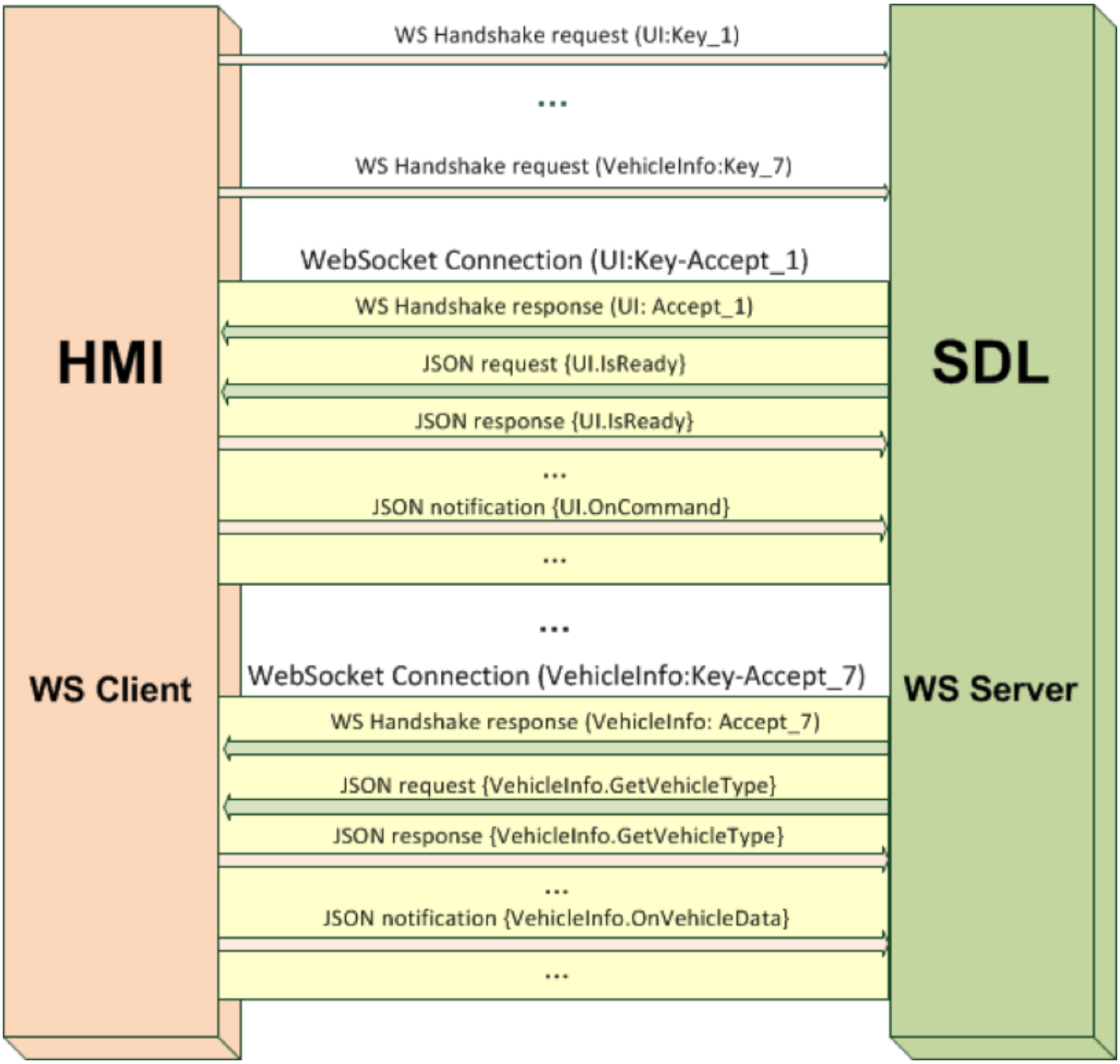
REQUEST

```
{
 "id": 700,
 "jsonrpc": "2.0",
 "method": "MB.registerComponent",
 "params": {
  "componentName": "VehicleInfo"
 }
}
```

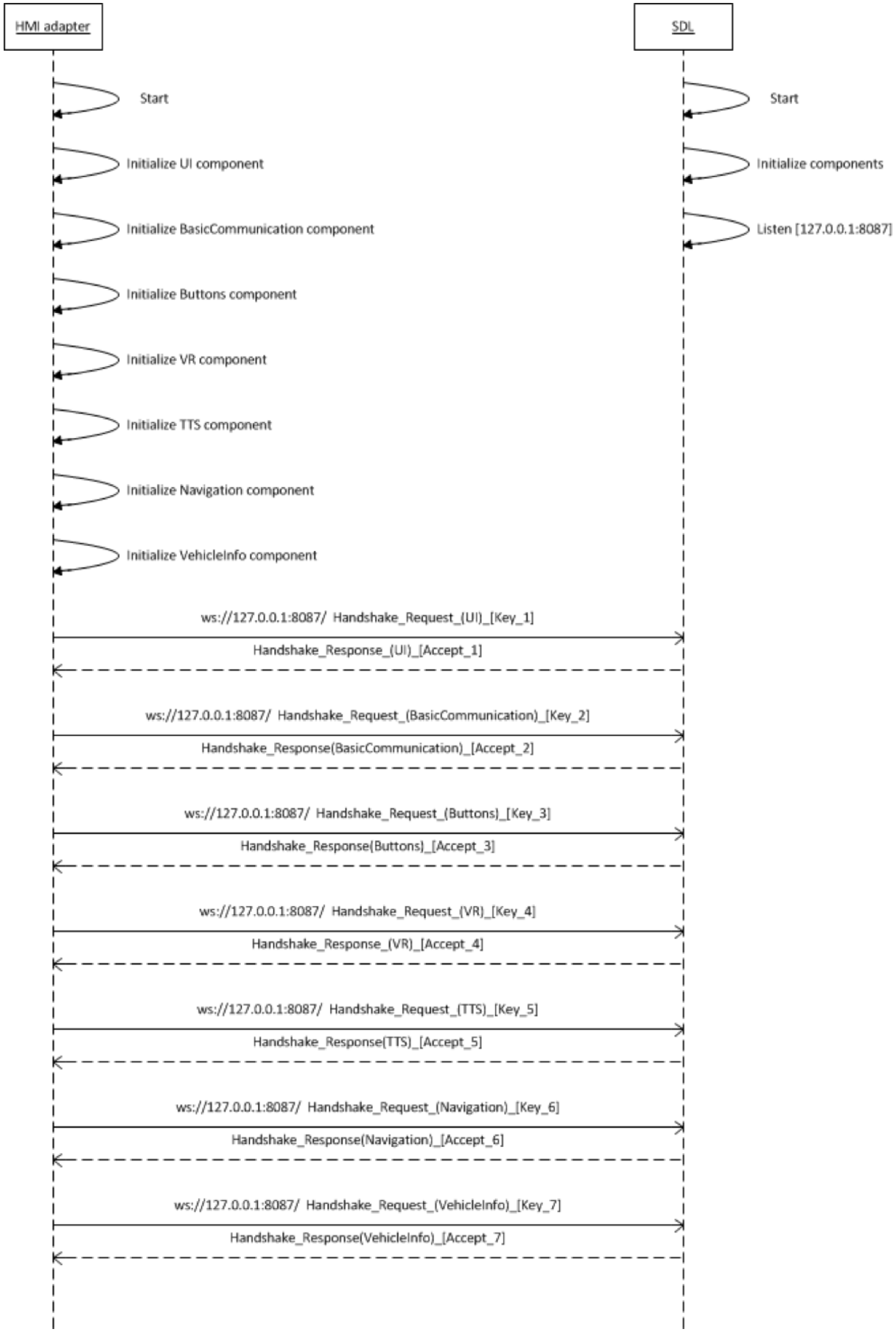
RESPONSE

```
{
 "id": 700,
 "jsonrpc": "2.0",
 "result": 7000
}
```

WEBSOCKET CONNECTION DIAGRAM



HMI ADAPTER INITIALIZATION


HMI adapter
SDL
Start
Start
Initialize UI component
Initialize components
Initialize BasicCommunication component
Listen [127.0.0.1:8087]
Initialize Buttons component
Initialize VR component
Initialize TTS component
Initialize Navigation component
Initialize VehicleInfo component
ws://127.0.0.1:8087/ Handshake_Request_(UI)_[Key_1]
Handshake_Response_(UI)_[Accept_1]
ws://127.0.0.1:8087/ Handshake_Request_(BasicCommunication)_[Key_2]
Handshake_Response(BasicCommunication)_[Accept_2]
ws://127.0.0.1:8087/ Handshake_Request_(Buttons)_[Key_3]
Handshake_Response(Buttons)_[Accept_3]
ws://127.0.0.1:8087/ Handshake_Request_(VR)_[Key_4]
Handshake_Response_(VR)_[Accept_4]
ws://127.0.0.1:8087/ Handshake_Request_(TTS)_[Key_5]
Handshake_Response(TTS)_[Accept_5]
ws://127.0.0.1:8087/ Handshake_Request_(Navigation)_[Key_6]
Handshake_Response(Navigation)_[Accept_6]
ws://127.0.0.1:8087/ Handshake_Request_(VehicleInfo)_[Key_7]
Handshake_Response(VehicleInfo)_[Accept_7]

JSON Message Format

This section describes the message structure for communication between your HMI and SDL. The JSON RPC 2.0 format is taken as a basis.

From this point forward the actors for exchanging messages will be considered:
* Client - can send requests and notifications
* Server - can provide responses to requests from a Client and send notifications

Request

An RPC call is represented by sending a Request object to a Server. The Request object has the following properties

PROPERTY	DESCRIPTION
"id"	An identifier established by the Client. This value must be of unsigned int type in the frames of communication between your HMI and SDL. The value should never be Null. If "id" is not included the message is assumed to be a notification and the receiver should not respond.
"jsonrpc"	A string specifying the version of JSON RPC protocol being used. Must be exactly "2.0" currently in all versions of SDL.
"method"	A String containing the information of the method to be invoked. The format is `[componentName].[methodName]`.
"params"	A structured value that holds the parameter values to be used during the invocation of the method. This property may be omitted.

Example Requests

REQUEST WITH NO PARAMETERS

```
{
 "id": 125,
 "jsonrpc": "2.0",
 "method": "Buttons.GetCapabilities"
}
```

REQUEST WITH PARAMETERS


```
{
  "id": 92,
  "jsonrpc": "2.0",
  "method": "UI.Alert",
  "params": {
    "alertStrings": [
      {
        "fieldName": "alertText1",
        "fieldText": "WARNING"
      },
      {
        "fieldName": "alertText2",
        "fieldText": "Adverse Weather Conditions Ahead"
      }
    ],
    "duration": 4000,
    "softButtons": [
      {
        "type": "TEXT",
        "text": "OK",
        "softButtonID": 697,
        "systemAction": "STEAL_FOCUS"
      }
    ],
    "appID": 8218
  }
}
```


Notification

A notification is a request object without an `id` property. For all the other properties, see the Request Section

The receiver should not reply to a notification, ie no response object needs to be returned to the client upon receipt of a notification.

Example Notifications

NOTIFICATION WITH NO PARAMETERS

```
{
 "jsonrpc": "2.0",
 "method": "UI.OnReady"
}
```

NOTIFICATIONS WITH PARAMETERS

```
{
 "jsonrpc": "2.0",
 "method": "BasicCommunication.OnAppActivated",
 "params": {
  "appID": 6578
 }
}

{
 "jsonrpc": "2.0",
 "method": "Buttons.OnButtonPress",
 "params": {
  "mode": "SHORT",
  "name": "OK"
 }
}
```

Response

On receipt of a request message, the server must reply with a response. The response is expressed as a single JSON Object with the following properties.

PROPERTY	DESCRIPTION
"id"	Required property which must be the same as the value of the associated request object. If there was an error in detecting the id in the request object, this value must be Null
"jsonrpc"	Must be exactly "2.0"
"result"	Required on Success. Must not exist if there was an error invoking the method. The result property must contain a `method` field which is the same as the corresponding request, a `code` field with 0 to indicate success. No other result codes can be sent in the response object. The result property may also include additional properties as defined in the HMI_API.

Example Responses

RESPONSE WITH NO PARAMETERS


```
{
 "id": 167,
 "jsonrpc": "2.0",
 "result": {
  "code": 0,
  "method": "UI.Alert"
 }
}
```

RESPONSE WITH PARAMETERS


```
{
 "id": 125,
 "jsonrpc": "2.0",
 "result": {
  "capabilities" : [
   {
    "longPressAvailable" : true,
    "name" : "PRESET_0",
    "shortPressAvailable" : true,
    "upDownAvailable" : true
   },
   {
    "longPressAvailable" : true,
    "name" : "TUNEDOWN",
    "shortPressAvailable" : true,
    "upDownAvailable" : true
   }
  ],
  "presetBankCapabilities": {
   "onScreenPresetsAvailable" : true
  },
  "code" : 0,
  "method" : "Buttons.GetCapabilities"
 }
}
```


Error Response

MUST

When an RPC encounters an error, the response object must contain the `error` property instead of the `result` property.

The error object has the following members:

PROPERTY	DESCRIPTION
"id"	Required to be the same as the value of "id" in the corresponding Request object. If there was an error in detecting the id of the request object, then this property must be Null
"jsonrpc"	Must be exactly "2.0"
"error"	Required on error. Must not exist if there was no error triggered during invocation. The error field must contain a `code` field with the value that indicates the error type that occurred (TODO: Result Enumeration section 5.1.1), a `message` field containing the string that provides a short description of the error, and a `data` field that must contain the `method` from the original request.

Examples

RESPONSE WITH ERROR

```
{
  "id": 103,
  "jsonrpc": "2.0",
  "error": {
    "code": 13,
    "message": "One of the provided IDs is not valid",
    "data": {
      "method": "VehicleInfo.GetDTCs"
    }
  }
}
```

Basic Communication

This section describes the messaging and HMI requirements between SDL and the HMI's basic communication component.

UpdateDeviceList

[TODO: looking for feedback on how to do format this section]

- Method: BasicCommunication.UpdateDeviceList
- Sender: SDL
- Purpose: Update HMI's list of known, connected devices

The `UpdateAppList` request is sent after SDL has found a new device over one of the available transports.

NOTE

The SDL's default Transport Manager (TM) and Transport Adapters (TA) behave in the following way.

1. Bluetooth Transport
    - TM performs a periodic search over Bluetooth.
    - Once the device is found, SDL starts the procedure of searching for SDL Enabled Applications on the device.
    - SDL Sends `BasicCommunication.UpdateDeviceList` with the name and id of the discovered device to the HMI.
    - SDL needs to receive `OnDeviceChosen(deviceInfo)` or `OnFindApplications(deviceInfo)` from the HMI to allow registering applications which are running on the connected device.
    - SDL allows registration of applications on the named device and sends `BasicCommunication.OnAppRegistered` to the HMI
    - SDL sends `BasicCommunication.UpdateDeviceList` to HMI when iOS device was connected over Bluetooth with applications registered and running on SDL and was also connected over USB.
2. USB Transport (AOA)
    - SDL learns about a new device connected over USB (A notification to the TA from the OS)
    - SDL Sends `BasicCommunication.UpdateDeviceList` with the name, id, and TransportType (AOA, iOS) of the discovered device to the HMI.
    - SDL sends `BasicCommunication.OnAppRegistered` to the HMI and does not wait for `OnDeviceChosen` or `OnFindApplications` notifications
3. Wifi Transport
    - SDL learns about a new device connected over Wifi
    - SDL sends `BasicCommunication.UpdateDeviceList` with the name and id of the discovered device to the HMI
    - SDL sends `BasicCommunication.OnAppRegistered` to the HMI and does not wait for `OnDeviceChosen` or `OnFindApplications` notifications

4. Cloud App Websockets
    - The application manager queries the policy table for cloud app policy entries
    - For each policy table entry, the websocket transport adapter creates a cloud device for each cloud application.
    - SDL sends `BasicCommunication.UpdateDeviceList` with the names, IDs, and **transportType: CLOUD_WEBSOCKET** for each cloud device

Behavior

MUST

1. Update list of connected devices (i.e. store the provided information)
2. Use the information when providing `OnDeviceChosen` and `OnFindApplications` notifications
3. Provide the user with the possibility of choosing among the found devices through either display, voice recognition or both.

REQUEST

PARAMETERS

NAME	TYPE	MANDATORY	ADDITIONAL
deviceList	Common.DeviceInfo	true	array: true minsize: 0 maxsize: 100

RESPONSE

PARAMETERS

This RPC has no additional parameter requirements

Example Request

```
{
 "id" : 64,
 "jsonrpc" : "2.0",
 "method" : "BasicCommunication.UpdateDeviceList",
 "params" :
 {
  "deviceList" :
  [
   {
    "name" : "Jerry`s Phone",
    "id" : 3
   },

   {
    "name" : "XT910",
    "id" : 4
   }
  ]
 }
}
```

Example Response

```
{
 "id" : 64,
 "jsonrpc" : "2.0",
 "result" : {
  "code" : 0,
  "method" : "BasicCommunication.UpdateDeviceList"
 }
}
```

Example Error

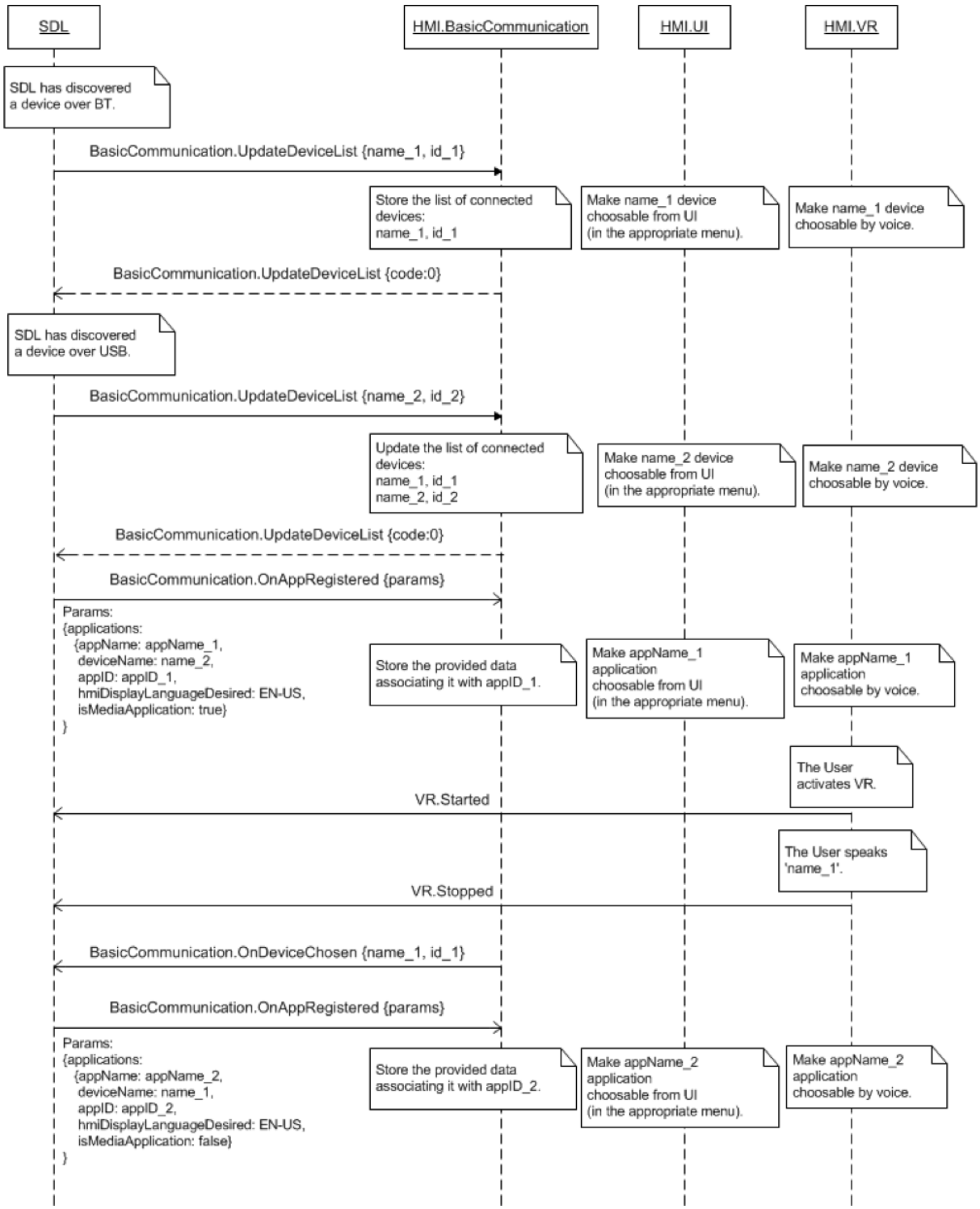
```
{
 "id" : 64,
 "jsonrpc" : "2.0",
 "error" : {
  "code" : 11,
  "message" : " The data sent is invalid.",
  "data" : {
   "method" : "BasicCommunication.UpdateDeviceList"
  }
 }
}
```

Sequence Diagrams

SEQUENCE DIAGRAM

UpdateDeviceList after SDL discovers device over BT or USB

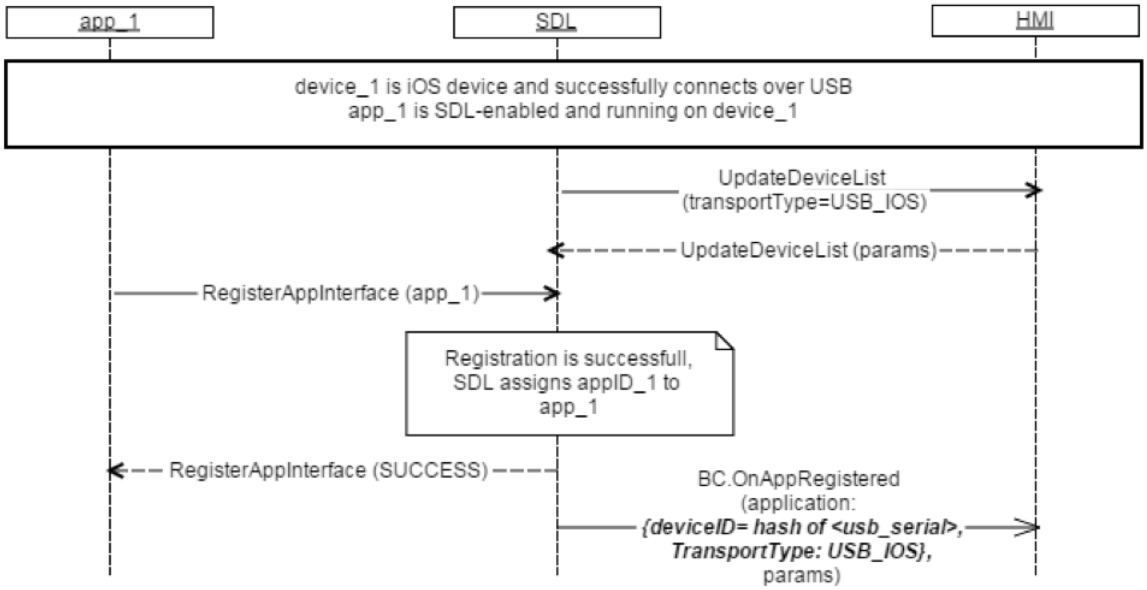
View Diagram



SEQUENCE DIAGRAM

UpdateDeviceList iOS

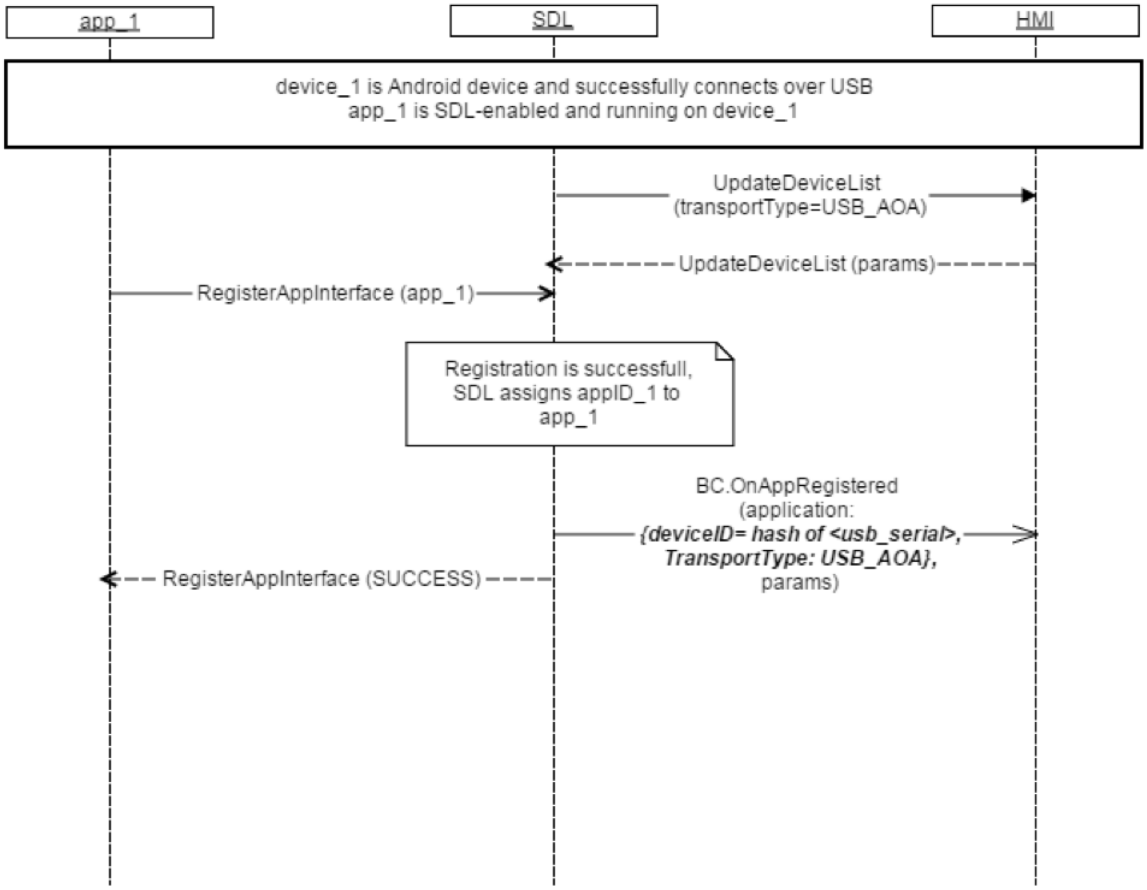
View Diagram



SEQUENCE DIAGRAM

UpdateDeviceList USB AOA

View Diagram


app_1
SDL
HMI
device_1 is Android device and successfully connects over USB
app_1 is SDL-enabled and running on device_1
UpdateDeviceList
(transportType=USB_AOA)
UpdateDeviceList (params)
RegisterAppInterface (app_1)
Registration is successfull,
SDL assigns appID_1 to
app_1
BC.OnAppRegistered
(application:
{deviceID= hash of <usb_serial>,
TransportType: USB_AOA},
params)
RegisterAppInterface (SUCCESS)

ActivateApp

NOTE

The activated application is assumed to receive access to the head unit's display, audio and control system which are:
* Supported by the head unit and
* Confirmed to be supported via responses to `IsReady` requests sent by SDL for the corresponding component

In the case of a successful HMILevel resumption, SDL will first assign a 'default_hmi' level from policies with a 3 second timeout, before resuming the application to either FULL or LIMITED.

If the HMI receives `ActivateApp` it can be assumed that the HMI has also already been provided with detailed information about the application through the OnAppRegistered RPC

MAY

This request may be sent:
* After SDL restores the application's state saved during a previous ignition off

Behavior

MUST

1. Activate the application on the HMI
    - Display UI.Show related parameters associated with the named `appID` in the case they were previously requested within ignition cycle
    - Display the corresponding template in the case one was previously requested by UI.SetDisplayLayout for that application
    - Apply UI.SetGlobalProperties associated with the named `appID`, if any
    - Apply UI.AddSubMenu associated with the named `appID`, if any
2. Make VR commands accessible from previous VR.AddCommand for the named `appID` during the same ignition cycle
3. Apply TTS.SetGlobalProperties associated with the `appID` in case previously requested since application registration
4. Assign priority based on the `priority` parameter received. If the parameter is omitted, the HMI must assign a priority of `NONE` by default
5. Respond with `SUCCESS` result code after the previous requirements have been met
6. Set up the application as the active audio source if it is a media or navigation type application

REQUEST

PARAMETERS

NAME	TYPE	MANDATORY	ADDITIONAL
appID	Integer	true	
priority	Common.AppPriority	false	
level	Common.HMILevel	false	

RESPONSE

PARAMETERS

This RPC has no additional parameter requirements

Example Request


```
{
 "id" : 47,
 "jsonrpc" : "2.0",
 "method" : "BasicCommunication.ActivateApp",
 "result" :
 {
  "appID" : 65368
 }
}
```


Example Response


```
{
 "id" : 47,
 "jsonrpc" : "2.0",
 "result" :
 {
  "code" : 0,
  "method" : "BasicCommunication.ActivateApp"
 }
}
```

Example Error

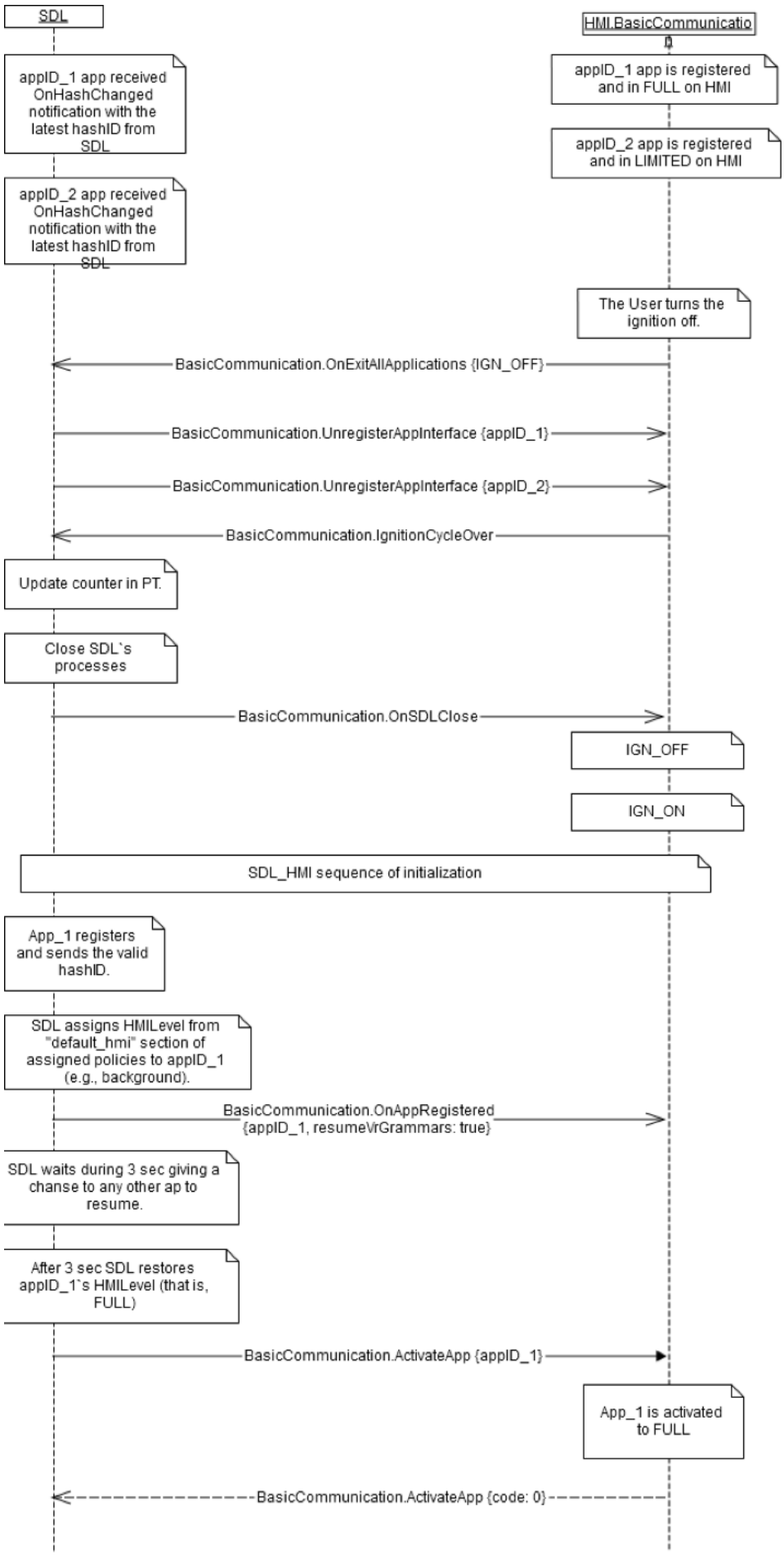

```
{
 "id" : 47,
 "jsonrpc" : "2.0",
 "error" :
 {
  "code" : 13,
  "message" : "One of the provided IDs is not valid.",
  "data" :
  {
   "method" : "BasicCommunication.ActivateApp"
  }
 }
}
```


Sequence Diagrams

SEQUENCE DIAGRAM

Activate App after successful Resumption

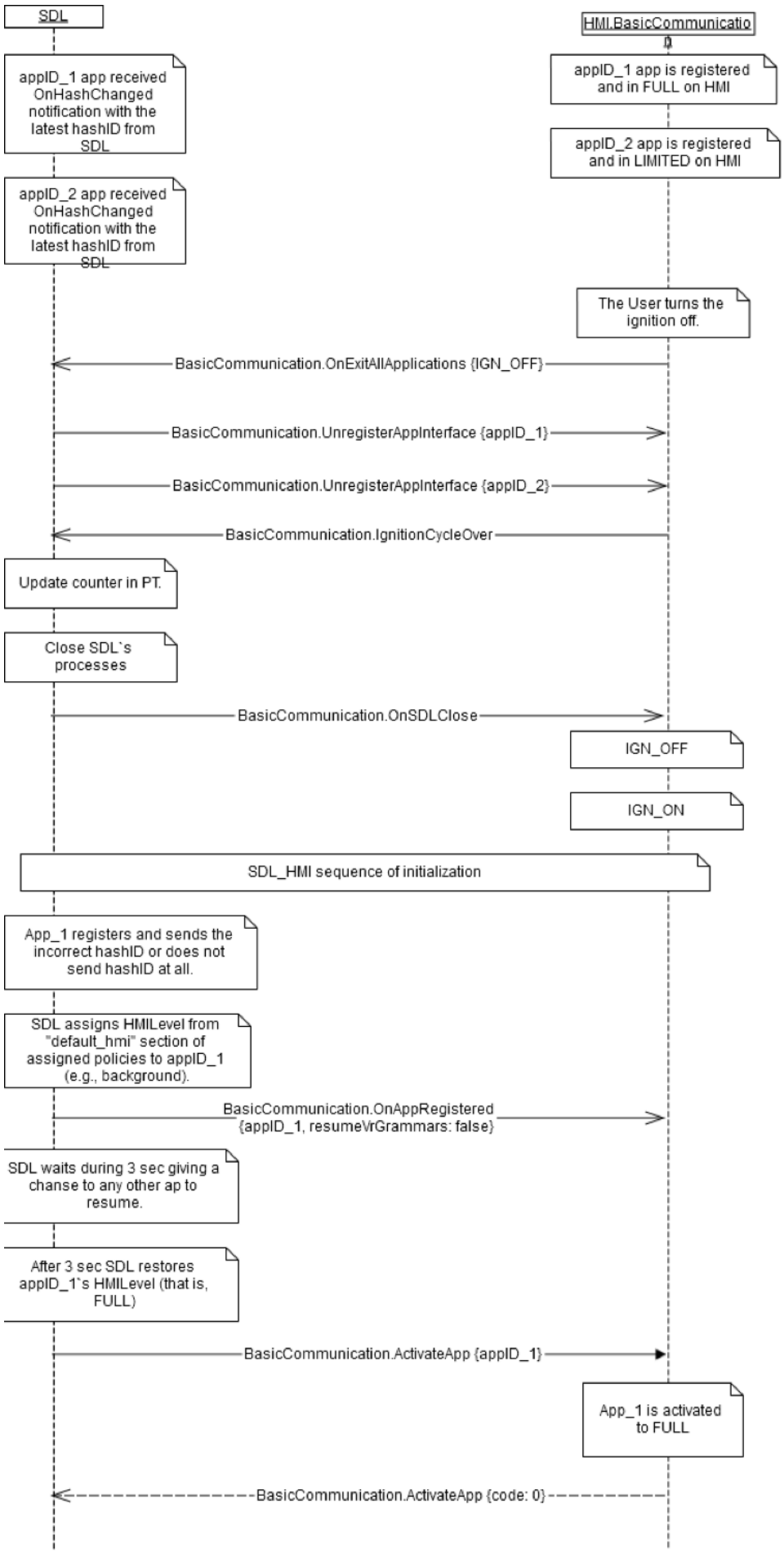
View Diagram


SDL
HMI.BasicCommunicatio
appID_1 app received OnHashChanged notification with the latest hashID from SDL
appID_1 app is registered and in FULL on HMI
appID_2 app is registered and in LIMITED on HMI
appID_2 app received OnHashChanged notification with the latest hashID from SDL
The User turns the ignition off.
BasicCommunication.OnExitAllApplications {IGN_OFF}
BasicCommunication.UnregisterAppInterface {appID_1}
BasicCommunication.UnregisterAppInterface {appID_2}
BasicCommunication.IgnitionCycleOver
Update counter in PT.
Close SDL`s processes
BasicCommunication.OnSDLClose
IGN_OFF
IGN_ON
SDL_HMI sequence of initialization
App_1 registers and sends the valid hashID.
SDL assigns HMILevel from "default_hmi" section of assigned policies to appID_1 (e.g., background).
BasicCommunication.OnAppRegistered {appID_1, resumeVrGrammars: true}
SDL waits during 3 sec giving a chanse to any other ap to resume.
After 3 sec SDL restores appID_1`s HMILevel (that is, FULL)
BasicCommunication.ActivateApp {appID_1}
App_1 is activated to FULL
BasicCommunication.ActivateApp {code: 0}

SEQUENCE DIAGRAM

Activate App after failed Resumption

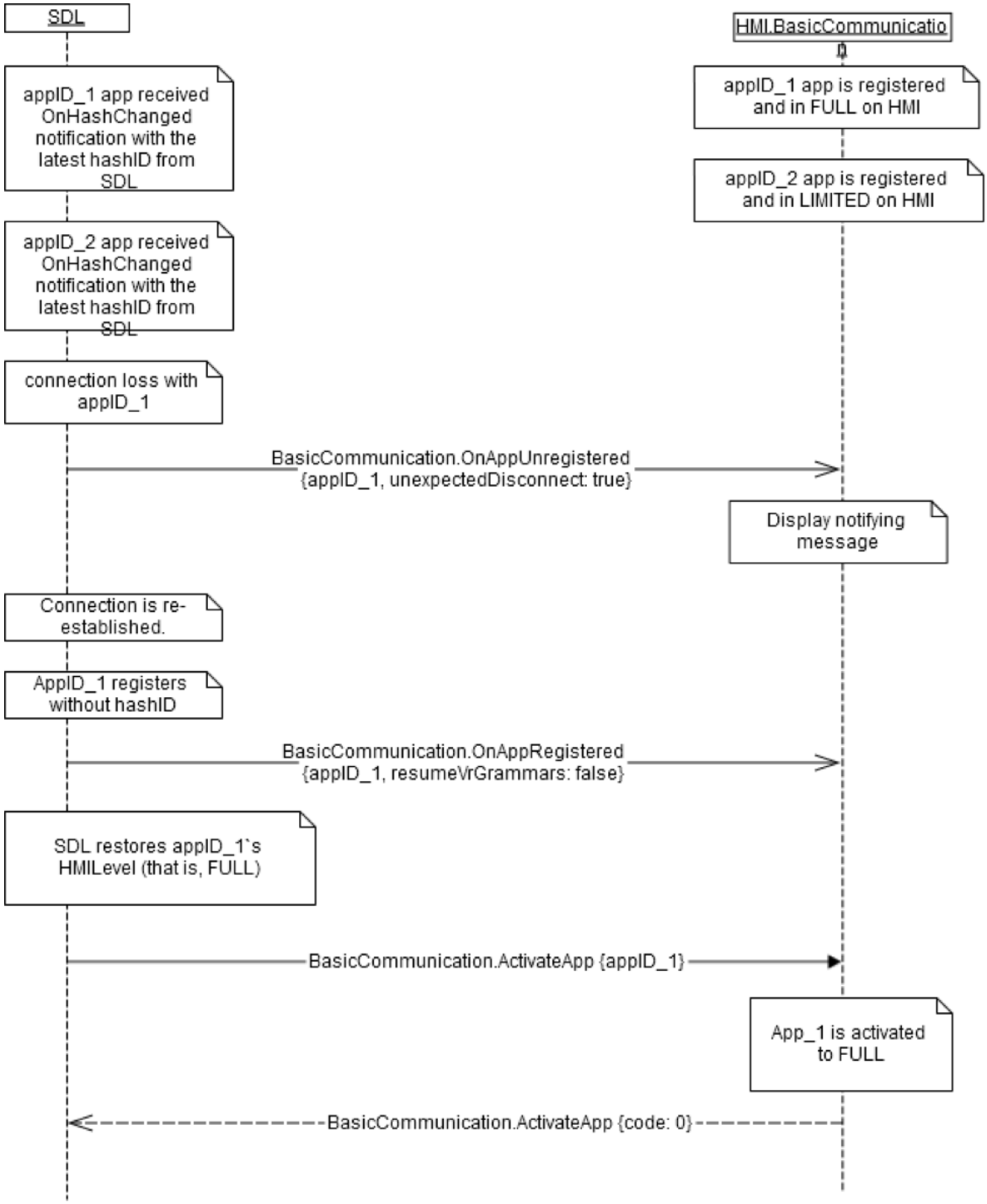
View Diagram


SDL
HMI.BasicCommunicatio
appID_1 app received OnHashChanged notification with the latest hashID from SDL
appID_1 app is registered and in FULL on HMI
appID_2 app is registered and in LIMITED on HMI
appID_2 app received OnHashChanged notification with the latest hashID from SDL
The User turns the ignition off.
BasicCommunication.OnExitAllApplications {IGN_OFF}
BasicCommunication.UnregisterAppInterface {appID_1}
BasicCommunication.UnregisterAppInterface {appID_2}
BasicCommunication.IgnitionCycleOver
Update counter in PT.
Close SDL`s processes
BasicCommunication.OnSDLClose
IGN_OFF
IGN_ON
SDL_HMI sequence of initialization
App_1 registers and sends the incorrect hashID or does not send hashID at all.
SDL assigns HMILevel from "default_hmi" section of assigned policies to appID_1 (e.g., background).
BasicCommunication.OnAppRegistered {appID_1, resumeVrGrammars: false}
SDL waits during 3 sec giving a chanse to any other ap to resume.
After 3 sec SDL restores appID_1`s HMILevel (that is, FULL)
BasicCommunication.ActivateApp {appID_1}
App_1 is activated to FULL
BasicCommunication.ActivateApp {code: 0}

SEQUENCE DIAGRAM

Activate App after Unexpected Disconnect

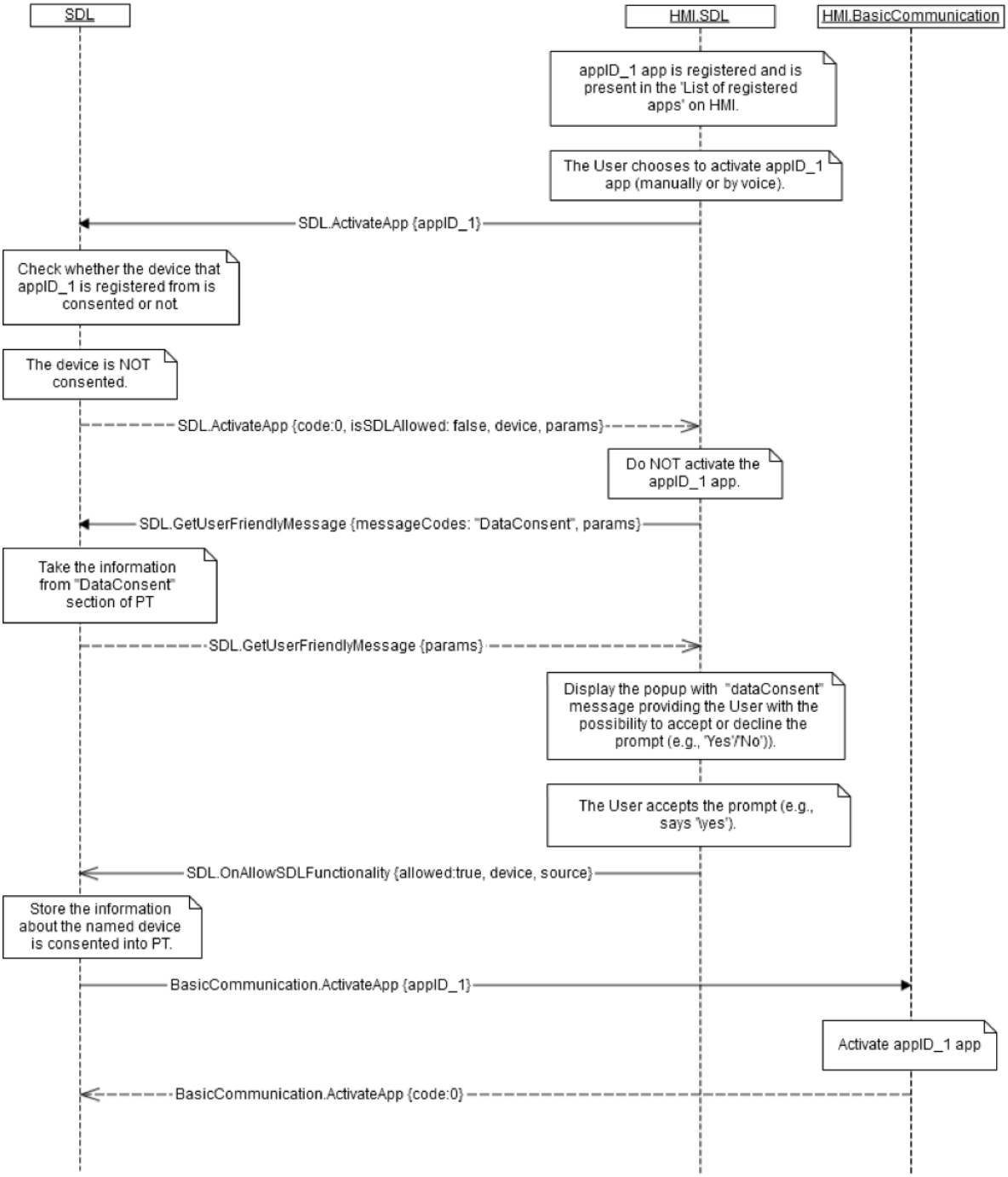
View Diagram



SEQUENCE DIAGRAM

Activate App after Accepted Data Consent Prompt

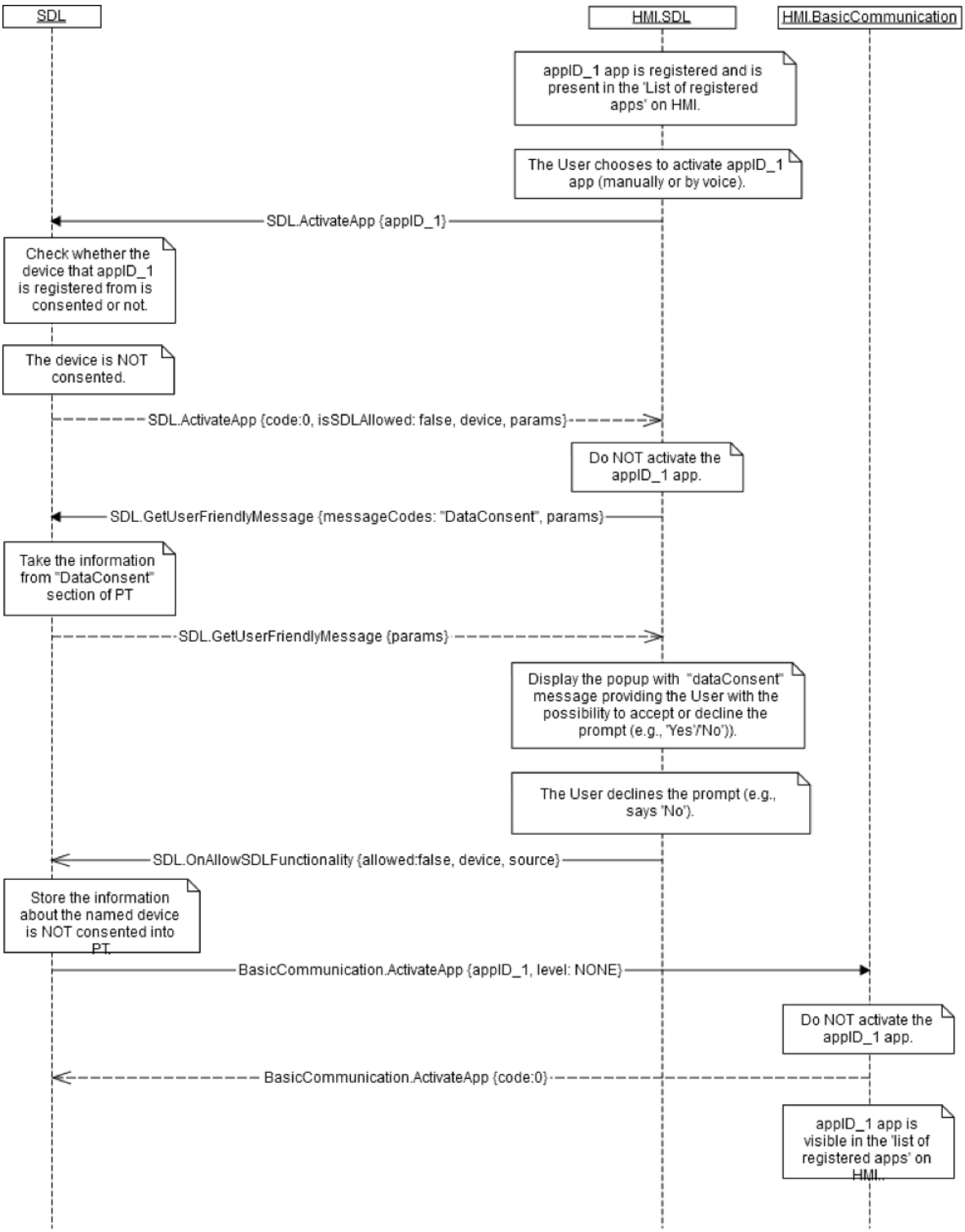
View Diagram



SEQUENCE DIAGRAM

Activate App after Failed Data Consent Prompt

View Diagram



MixingAudioSupported

Purpose: inform SDL whether the vehicle audio system has the ability to speak the TTS prompts or listen to an recognize VR commands while playing audio.

> NOTE
>
> SDL is able to get information about the audio mixing capabilities from smartDeviceLink.ini. If the request is ignored and there is no record for audio mixing capabilities in smartDeviceLink.ini, SDL will assume mixing audio is not supported.

Behavior

MUST

Check mixing audio capabilities and provided an accurate response.

REQUEST

PARAMETERS

NAME	TYPE	MANDATORY	ADDITIONAL

RESPONSE

PARAMETERS

NAME	TYPE	MANDATORY	ADDITIONAL
attenuatedSupported	Boolean	true	

Request Example

```
{
 "id" : 27,
 "jsonrpc" : "2.0",
 "method" : "BasicCommunication.MixingAudioSupported"
}
```

Response Example

```
{
 "id" : 27,
 "jsonrpc" : "2.0",
 "result" :
 {
  "attenuatedSupported" : true,
  "code" : 0,
  "method" : "BasicCommunication. MixingAudioSupported"
 }
}
```

Error Example

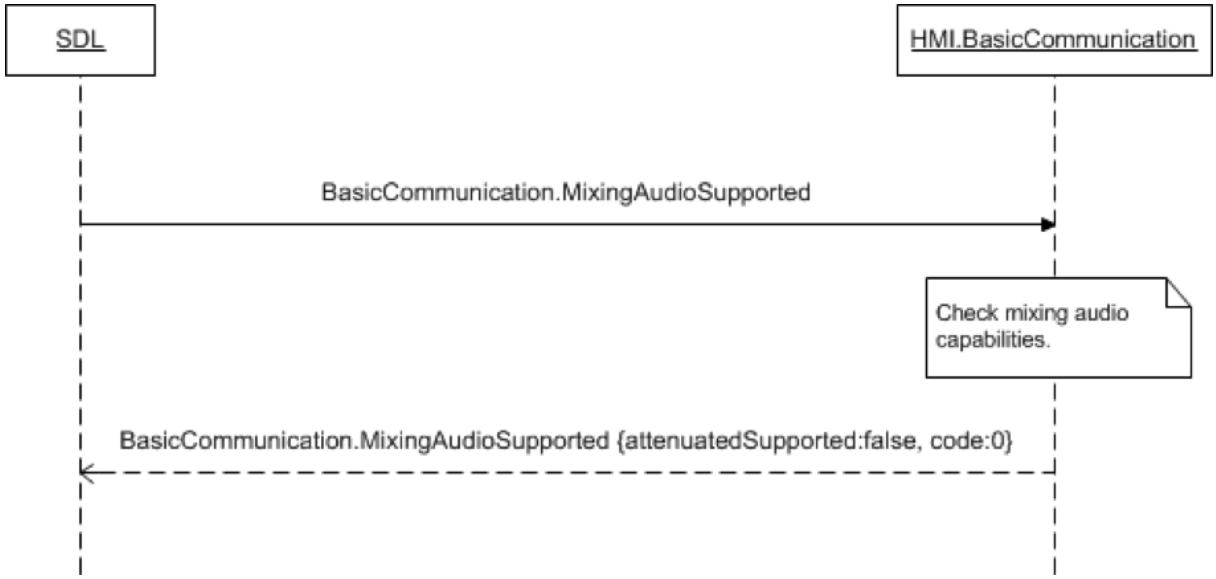
```
{
 "id" : 27,
 "jsonrpc" : "2.0",
 "error" :
 {
  "code" : 22,
  "message" : "An unknown error occurred",
  "data" :
  {
   "method" : "BasicCommunication.MixingAudioSupported"
  }
 }
}
```

Sequence Diagrams

SEQUENCE DIAGRAM

MixingAudioSupported Messaging

View Diagram



SystemRequest

- Type: Function
- Sender: SDL
- Provide the path to a system file that SDL has received from the mobile application

SDL sends SystemRequest to the HMI when SDL receives the SystemRequest RPC from a mobile application.

Behavior

> MUST
>
> Notify any relevant modules about the location of the file which was transferred.

NOTE

If the HMI does not respond to SDL's request within a specified timeout period (default of 10 seconds), SDL will return GENERIC_ERROR to the corresponding mobile apps request.

NOTE

SDL validates all SystemRequests sent from the mobile app and returns DISALLOWED if the app's system request contains a RequestType that is not allowed via Policies.
SDL sends the list of RequestTypes allowed by Policies via OnAppPermissionChanged, UpdateAppList, or OnSystemRequest RPCs.
If the HMI sends OnSystemRequest with a RequestType which is disallowed by the current policy table, SDL will ignore the notification.
If SDL sends a SystemRequest with `requestSubType` parameter to an older system, it would be rejected by the system with a response of INVALID_DATA.

REQUEST

PARAMETERS

NAME	TYPE	MANDATORY	ADDITIONAL
requestType	Common.RequestType	true	
requestSubType	String	false	maxlength="255"
fileName	String	true	minlength: 1 maxlength: 255
appID	String	false	minlength: 1 maxlength: 50

RESPONSE

PARAMETERS

This RPC has no additional parameter requirements

Example Request

```
{
 "id" : 59,
 "jsonrpc" : "2.0",
 "method" : "BasicCommunication.SystemRequest"
 "params" :
 {
  "requestType" : "FILE_RESUME",
  "fileName" : "/tmp/fs/mp/images/ivsu_cache/123.json",
  "appID" : 223
 }
}
```

Example Response

```
{
 "id" : 59,
 "jsonrpc" : "2.0",
 "result" :
 {
  "code" : 0,
  "method" : "BasicCommunication.SystemRequest"
 }
}
```

Example Error

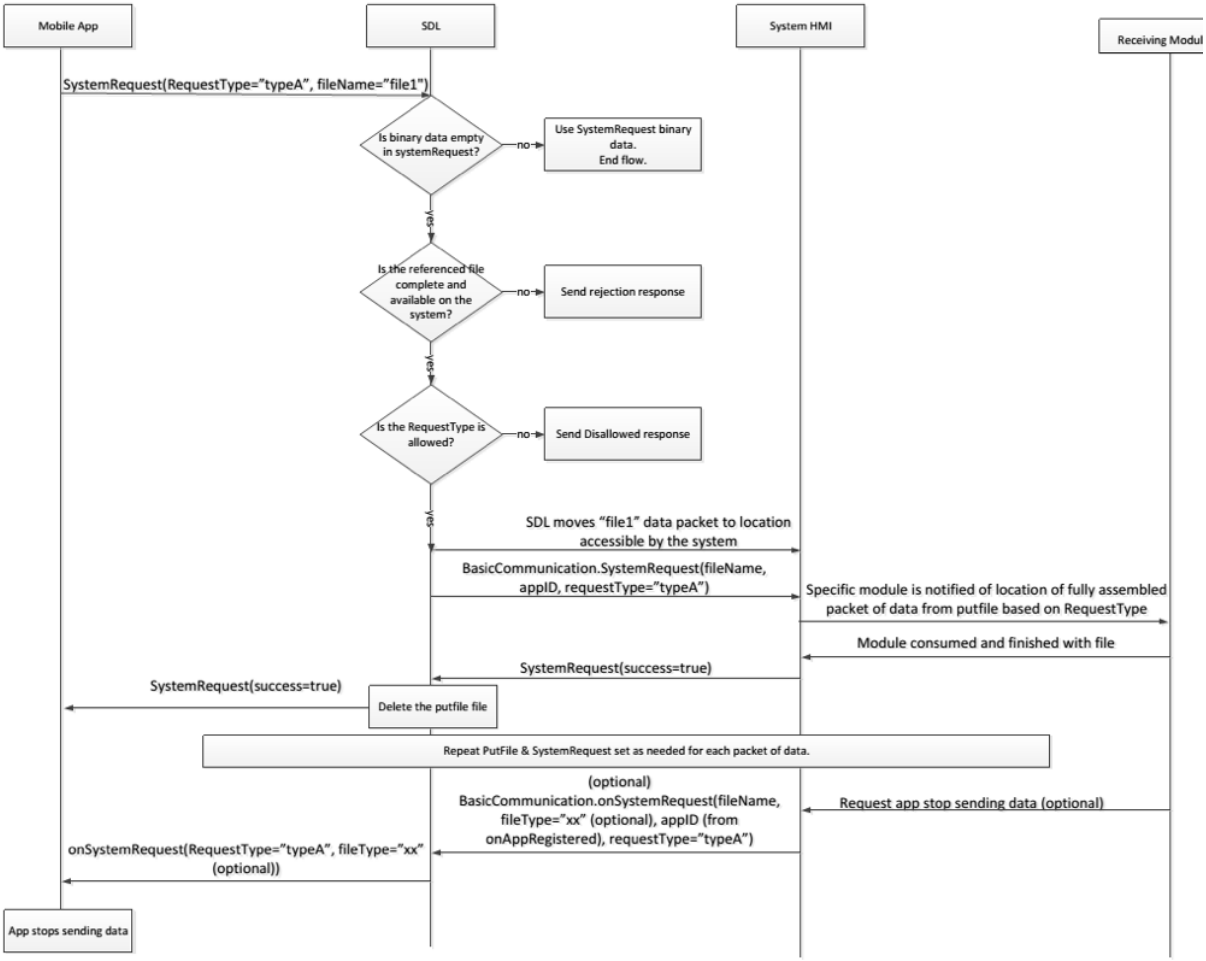

```
{
 "id" : 59,
 "jsonrpc" : "2.0",
 "error" :
 {
  "code" : 11,
  "message" : "Invalid data",
  "data" :
  {
   "method" : "BasicCommunication.SystemRequest"
  }
 }
}
```


Sequence Diagrams

SEQUENCE DIAGRAM

System Request Workflow

View Diagram



GetSystemInfo

- Type: Function
- Sender: SDL
- Purpose: Get head unit system information

MUST

The HMI must respond to GetSystemInfo and provide the current head unit software version, language, and country code.

NOTE

When SDL starts, it sends GetSystemInfo to update the policies database with the latest system data. SDL Stores the ccpu_version when responding to RegisterAppInterfaceRequests.

REQUEST

PARAMETERS

NAME	TYPE	MANDATORY	ADDITIONAL

RESPONSE

PARAMETERS

NAME	TYPE	MANDATORY	ADDITIONAL
ccpu_version	String	true	maxlength: 500
language	Common.Language	true	
wersCountryCode	String	true	maxlength: 500

Example Request

```
{
 "id" : 47,
 "jsonrpc" : "2.0",
 "method" : "BasicCommunication.GetSystemInfo"
}
```

Example Response


```
{
 "id" : 47,
 "jsonrpc" : "2.0",
 "result" : {
  "code" : 0,
  "method" : "BasicCommunication.GetSystemInfo",
  "params" : {
     "ccpu_version" : "12.1.3",
     "language" : "EN-US",
     "wersCountryCode" : "WAEGB"
     }
  }
}
```


Example Error

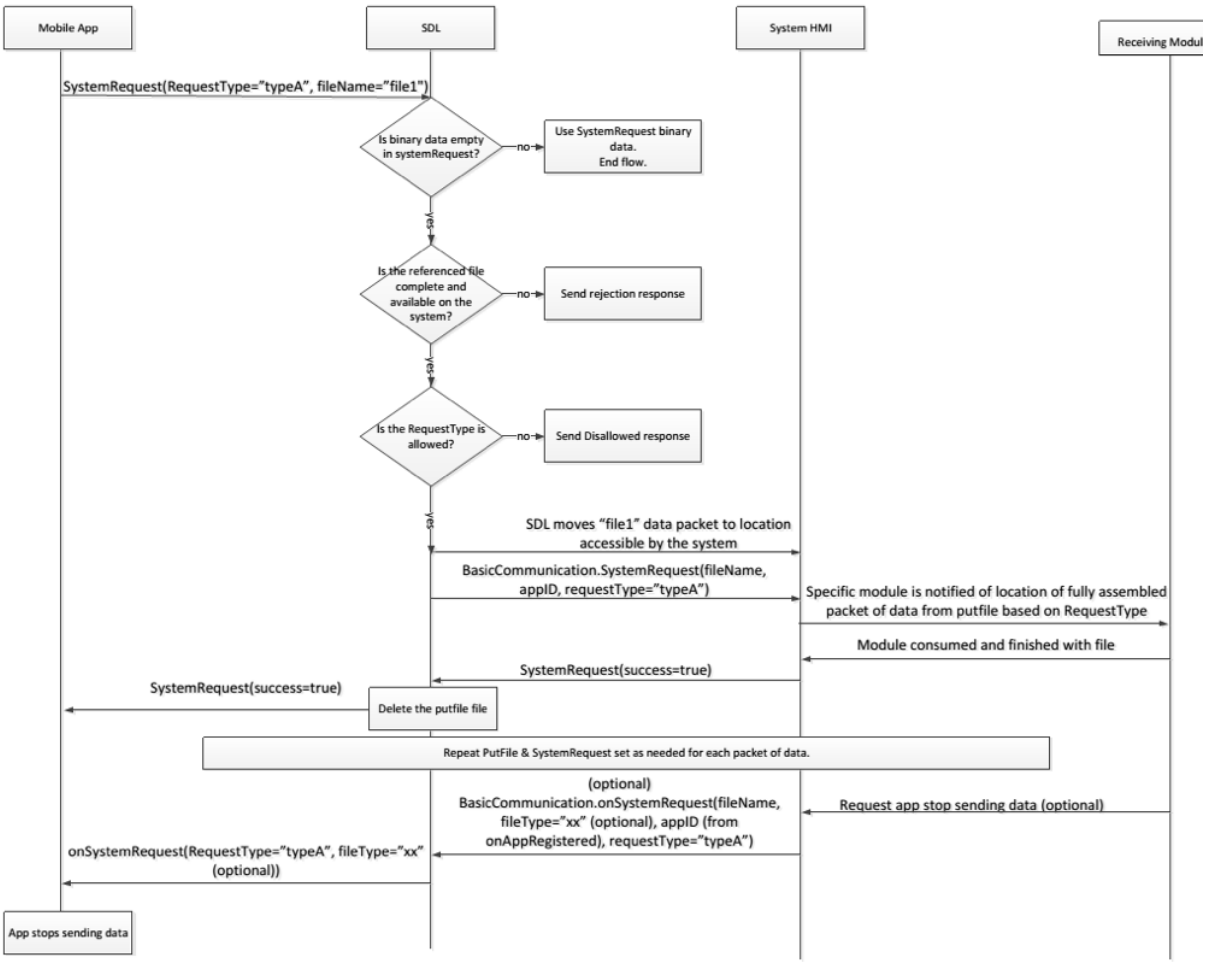

```
{
 "id" : 47,
 "jsonrpc" : "2.0",
 "error" :
 {
  "code" : 9,
  "message" : " Error in fetching system information ",
  "data" :
  {
   "method" : "BasicCommunication.GetSystemInfo"
  }
 }
}
```


Sequence Diagrams

SEQUENCE DIAGRAM

GetSystemInfo

View Diagram



OnReady

- Type: Notification
- Sender: HMI
- Purpose: Inform SDL about readiness to communicate

`OnReady` is the first message which begins the SDL-HMI communication after the WebSocket transports are established.

MUST

The HMI must send this notification after the connection is established and the HMI is ready for communication in order to communicate with SDL.

Example Notification

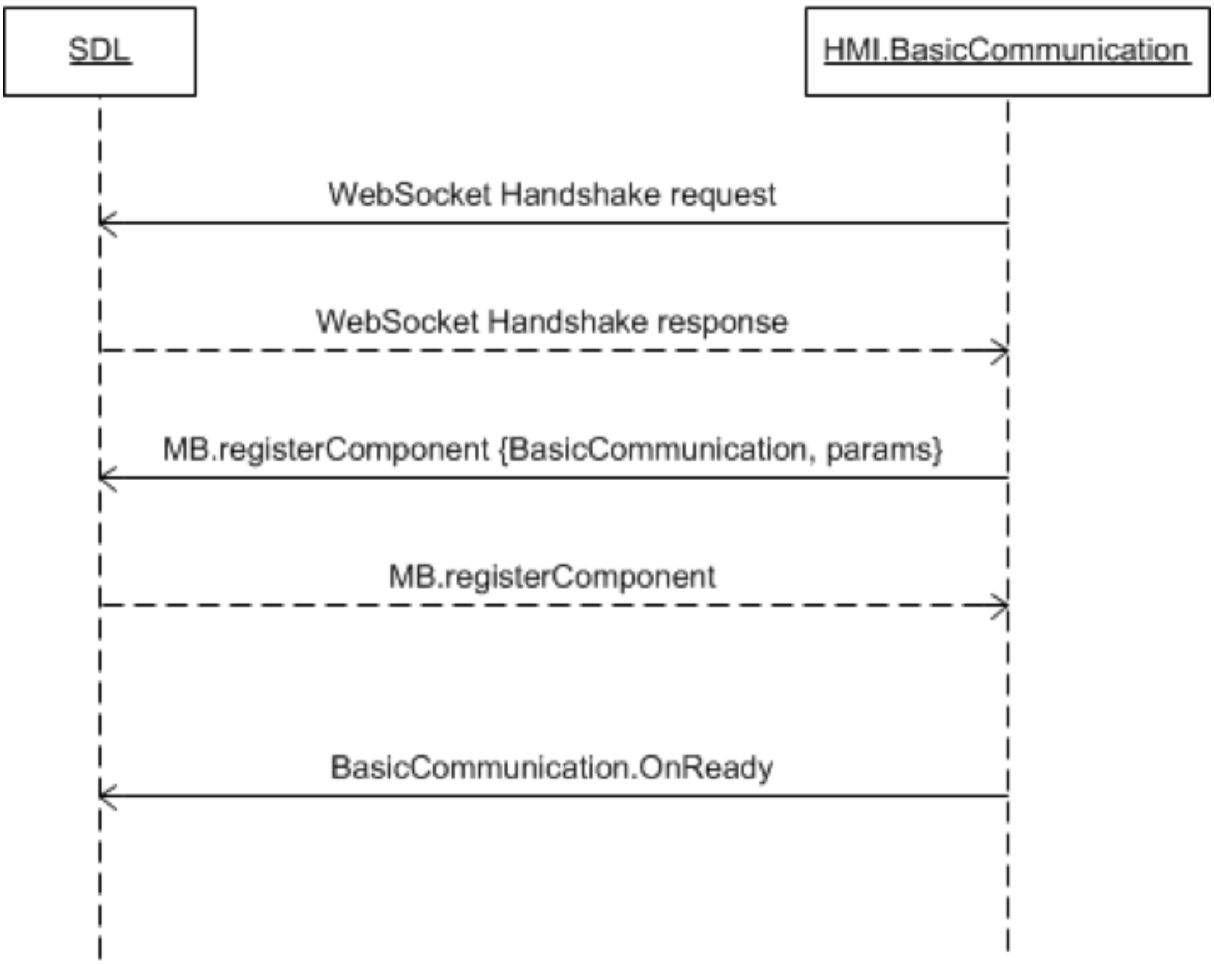
```
{
  "jsonrpc" : "2.0",
  "method" : "BasicCommunication.OnReady"
}
```

Sequence Diagrams

SEQUENCE DIAGRAM

OnReady WebSocket

View Diagram



JSON EXAMPLE NOTIFICATION

OnStartDeviceDiscovery

- Type: Notification

- Sender: HMI
- Purpose: Initiate device search

On receipt of this notification SDL starts searching for devices on all available transports. Afterwards, SDL provides the search results via UpdateDeviceList.

NOTE

This RPC tells SDL to initiate a new device search. The OnUpdateDeviceList RPC asks SDL for the results of the most recent device query.

Example Notification

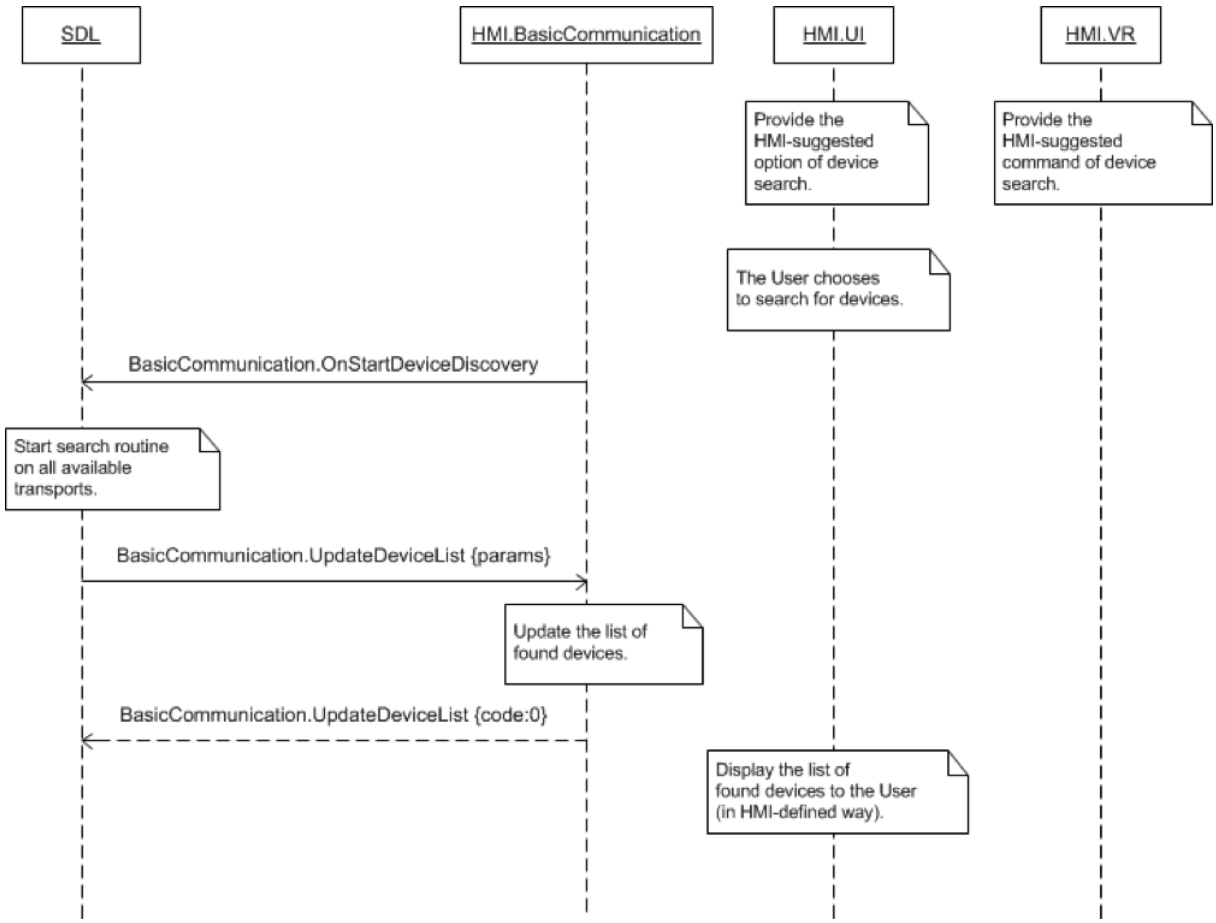
```
{
  "jsonrpc" : "2.0",
  "method" : "BasicCommunication.OnStartDeviceDiscovery"
}
```

Sequence Diagrams

SEQUENCE DIAGRAM

Starting Device Discovery

View Diagram



JSON EXAMPLE NOTIFICATION

OnDeviceChosen

Type

Notification

Sender

HMI

Purpose

Open the connection with the named device

On Receipt of this notification SDL will open the connected with the named device and start sessions with the compatible SDL applications running on the device.

MUST

1. Provide the user with the possibility to choose among the list of known devices
2. Send this RPC when the user has selected the device

NOTE

The list of known devices is provided to the HMI in the UpdateDeviceList Request.

Notification

PARAMETERS

NAME	TYPE	MANDATORY	ADDITIONAL
deviceInfo	Common.DeviceInfo	true	

Example Notification

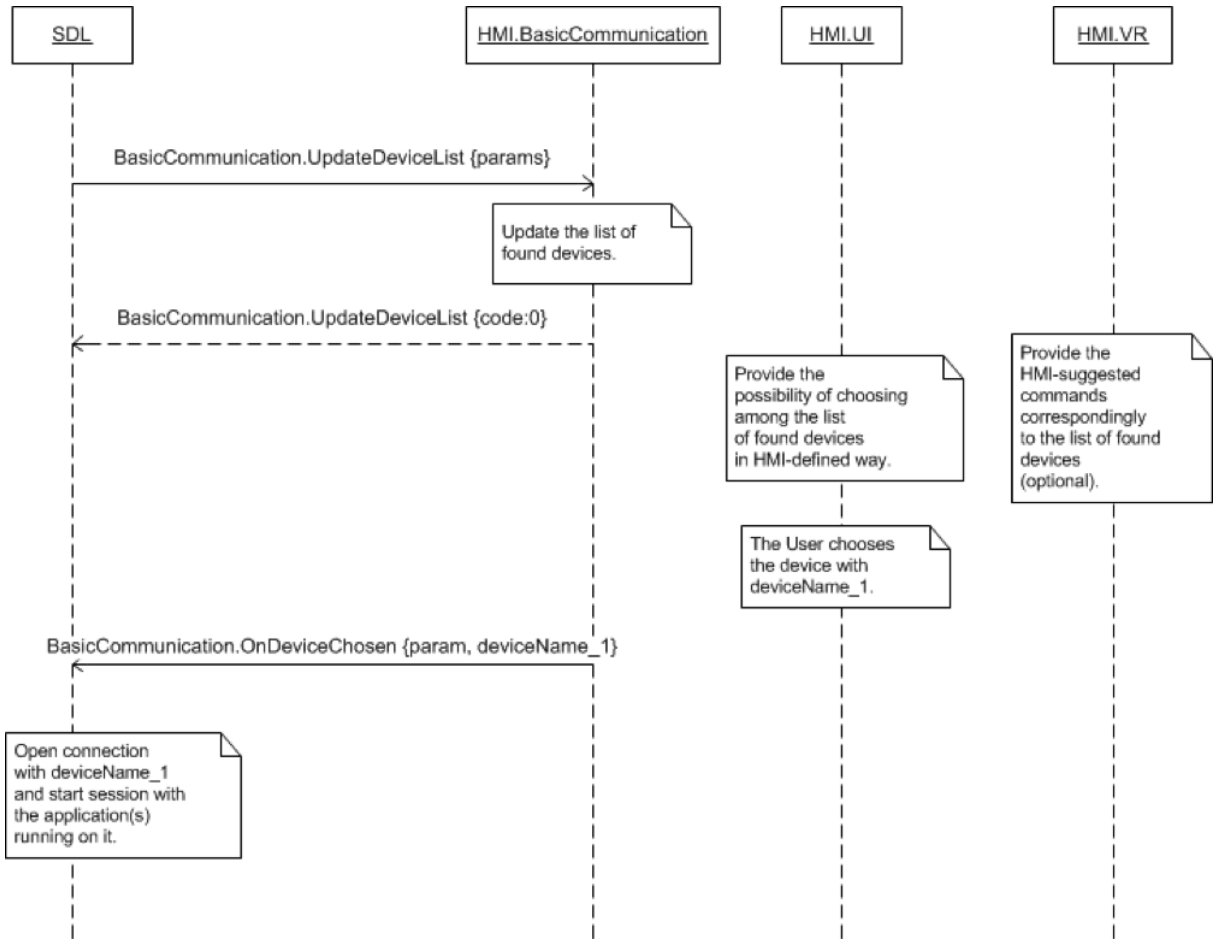
```
{
 "jsonrpc" : "2.0",
 "method" : "BasicCommunication.OnDevcieChosen",
 "params" :
 {
  "deviceInfo" :
  {
     "name" : "Jerry`s Phone",
     "id" : 3
  }
 }
}
```

Sequence Diagrams

SEQUENCE DIAGRAM

OnDeviceChosen

View Diagram



DialNumber

Type Function
Sender SDL

Purpose

SDL initiates a call to a specific phone number.

NOTE

SDL looks to see if the phone number entered is correct before passing to the HMI. The checks performed are:

1. Strip any characters except of 0-9 and * # , ;+ and pass resulting number to HMI.
2. Return INVALID_DATA to mobile side without transferring "number" to the HMI if "number" is empty after stripping invalid characters.
3. Return INVALID_DATA to mobile side without transferring "number" to HMI if the characters "/n" , "/t", or spaces are included in "number".

MUST

1. Show DialNumber pop-up on HMI with 2 buttons, "Call" and "Cancel".
2. Send the notification OnAppDeactivated(PHONECALL) to SDL when the phone call is started on the HMI. The notification must be sent to all applications that have active audio sources on the HMI.
3. Send the notification BC.OnOnPhoneCall(isActive:true) to SDL when the phone call is started on the HMI.
4. Send the notification BC.OnOnPhoneCall(isActive:false) to SDL when the phone call is ended on the HMI.
5. Always respond to BC.DialNumber with a response code. If the HMI does not respond, the mobile application will never get a response from from SDL because default timeouts do not apply to the DailNumber mobile API.

The request is considered to have been executed successfully only after the user presses the "Call" button included in the DialNumber dialog.

REQUEST

PARAMETERS

NAME	TYPE	MANDATORY	ADDITIONAL
number	String		maxlength: 40
appID	Integer	true	

RESPONSE

PARAMETERS

This RPC has no additional parameter requirements

Sequence Diagrams

SEQUENCE DIAGRAM

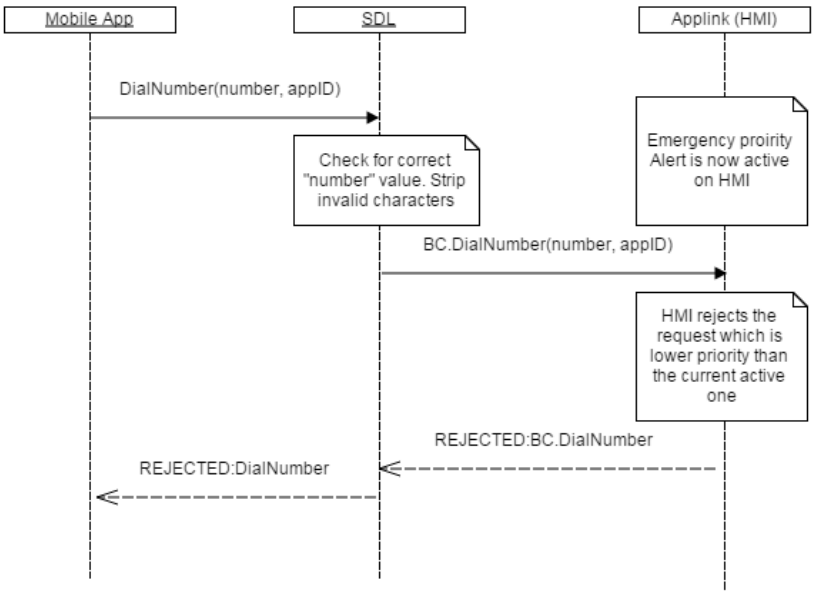
DialNumber Success

View Diagram

SEQUENCE DIAGRAM

|||
DialNumber Failed

View Diagram



Example Request

```
{
 "id" : 59,
 "jsonrpc" : "2.0",
 "method" : "BasicCommunication. DialNumber",
 "params" :
 {
     "number" : "*111#",
     "appID" : 65537
 }
}
```

Example Response

```
{
 "id" : 59,
 "jsonrpc" : "2.0",
 "result" :
 {
  "code" : 0,
  "method" : "BasicCommunication.DialNumber"
 }
}
```

Example Error

```
{
 "id" : 59,
 "jsonrpc" : "2.0",
 "error" :
 {
  "code" : 5,
  "message" : " HMI is busy with higher priority RPC ",
  "data" :
  {
   "method" : "BasicCommunication.DialNumber"
  }
 }
}
```

OnAppActivated

Type Notification
Sender HMI
Purpose Inform SDL that the user has chosen to activate an application.

SDL requires this notification to know if the user has requested for an application to be in focus and start operating, based on the HMI's functionality.

When `OnAppActivated` is received, SDL sends an `ActivateApp` request to confirm the named application may be activated.

MUST

1. Send an `OnAppActivated` notification when the user chooses an application on the HMI.
2. Wait for `ActivateApp` before the HMI can display the application related screen.

NOTE

`OnAppActivated` shows the user's intent to activate the named application.

`ActivateApp` is a request created by SDL that provides the permission for the application to access the HMI's functionality.

Notification

PARAMETERS

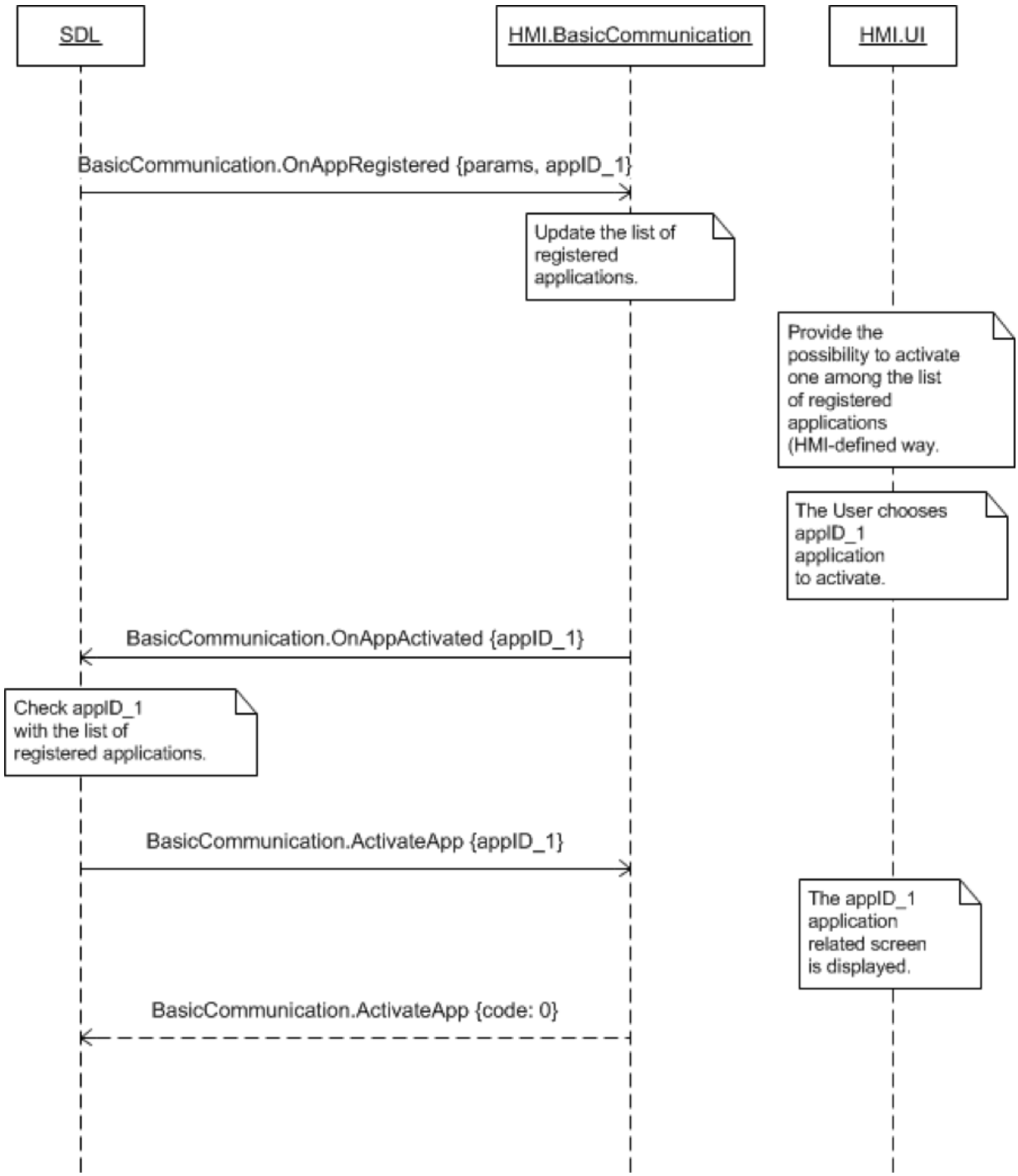
NAME	TYPE	MANDATORY	ADDITIONAL
appID	Integer	true	ID of the application to be activated on the HMI

Sequence Diagrams

SEQUENCE DIAGRAM

User Activates App

View Diagram


SDL
HMI.BasicCommunication
HMI.UI
BasicCommunication.OnAppRegistered {params, appID_1}
Update the list of registered applications.
Provide the possibility to activate one among the list of registered applications (HMI-defined way.
The User chooses appID_1 application to activate.
BasicCommunication.OnAppActivated {appID_1}
Check appID_1 with the list of registered applications.
BasicCommunication.ActivateApp {appID_1}
The appID_1 application related screen is displayed.
BasicCommunication.ActivateApp {code: 0}

JSON EXAMPLE NOTIFICATION

```
{
 "jsonrpc" : "2.0",
 "method" : "BasicCommunication.OnAppActivated",
 "params" :
 {
  "appID" : 65544
 }
}
```

OnAppDeactivated

Type Notification
Sender HMI
Purpose Inform the application it is no longer active on the HMI

SDL requires this notification in order to keep the mobile application from sending RPC's related to the HMI's functionality (e.g. adding commands for VR, starting an interaction with the user, speaking text via TTS, etc).

In the event a navigation application is in `FULL` and the HMI sends `OnAppDeactivated`, SDL must set the navigation application to `LIMITED` with the AudioStreamingState set to `AUDIBLE`, VideoStreamingState `STREAMABLE`.

In the event a media projection application is in `FULL` and the HMI sends `OnAppDeactivated` (user goes out from media app screen to HU menu), SDL must set the app to `LIMITED` with the AudioStreamingState set to `AUDIBLE`, VideoStreamingState `STREAMABLE`.

In the event a non media projection application is in `FULL` the HMI sends `OnAppDeactivated` (user goes out from media app screen to HU menu), SDL must set the app to `LIMITED` with the AudioStreamingState set to `NOT_AUDIBLE`, VideoStreamingState `STREAMABLE`.

NOTE

The information about the application (name, appID, etc) is provided by SDL via the RPCs `UpdateAppList` or `OnAppRegistered`. SDL ignores all invalid notifications which come from the HMI (invalid JSON, invalid data types/bounds, etc).

Notification

PARAMETERS

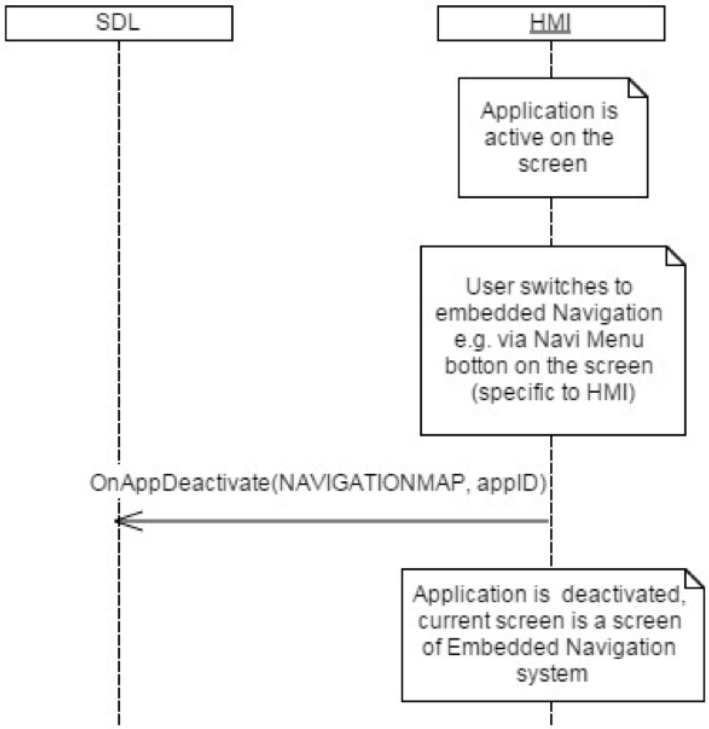
NAME	TYPE	MANDATORY	ADDITIONAL
appID	Integer	true	

Sequence Diagrams

SEQUENCE DIAGRAM

User Switches Apps

View Diagram



JSON EXAMPLE NOTIFICATION

```
{
 "jsonrpc" : "2.0",
 "method" : "BasicCommunication.OnAppDeactivated",
 "params" :
 {
  "appID" : 65544,
  "reason" : PHONECALL
 }
}
```

OnAppRegistered

Type Notification
Sender SDL
Purpose Update the HMI's list of registered applications and resume the Audio Source

SDL will send `OnAppRegistered`:

1. After an SDL-enabled application has registered successfully.
2. When the device was reconnected after an unexpected disconnect and a previously connected SDL application reregisters. SDL makes a decision if the data resumption process is applicable for the application.
3. When the HMI sends an `OnFindApplications` notification via the users request.

Regarding data resumption:

Data resumption means that an application may request to restore data used in the previous ignition cycle after an `Unexpected Disconnect`.

- For data resumption purposes, SDL must store application-related data such as commands, application global properties, and show data for the past three ignition cycles after an `Unexpected Disconnect` or `Ignition Off`. On the fourth `Ignition On`, SDL clears all corresponding application-related data used for resumption.

- HMI must store the VR grammar 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 `BC.OnResumeAudioSource`.

If the application resumes data successfully:

- SDL will provide `OnAppRegistered` with `resumeVrGrammars`:`true` to notify the HMI that `VRGrammars` must be resumed. On this event, the HMI must restore the application related `VRGgrammars` for the appID received via an `OnAppRegistered` notification.
- SDL must restore application-related data and send to the HMI after an `OnAppRegistered` notification:
    - `AddCommand`(Menu + VR)
    - `AddSubMenu`
    - `CreateInteractionChoiceSet`
    - `SetGlobalProperties`
    - `SubscribeButton`
    - `SubscibeVehicleData`

If the application does NOT resume data successfully:

- SDL will provide `OnAppRegistered` with `resumeVrGrammars`:`false` or no resume parameter at all.
- SDL 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 and the HMI should restart the cycle of collecting application data for resumption.

MUST

1. Update its list of registered applications.
2. Store the application data sent in the `applications` parameter.
3. Compile and store `VRGrammars` for the `vrSynonyms` parameter, and arrange them for the user to be able to use via voice recognition. Note: The VR commands to activate an application

must be accessible when viewing a different active application or the list of registered applications.
4. Provide the user with the possibility to choose an application among a list of registered applications.
5. Send an `OnAppActivated` notification to SDL when the user activates an app via the `UI` or `VR`.
6. Manage application events by priority. HMI gets proirity information from *OnAppRegistered*, *UpdateAppList*, *ActivateApp* HMI API.
7. HMI must set app icon and create the app title in the mobile apps list in case SDL provides `<icon>` via `OnAppRegistered` notification.
8. HMI must set default app icon in case SDL omits `<icon>` at `OnAppRegistered` notification.

NOTE

- If a device is connected over USB and registers an application, SDL will send `OnAppRegistered` with a hash of the usb serial number as the device id.
- If a device is connected over Bluetooth or Wi-Fi and registers an application, SDL will send `OnAppRegistered` with a hash of the device's mac address as the device id.
- When the application is registered for the first time (no records in PT) PoliciesManager should not initiate prompting the User about the event.
- If iOS device was connected over Bluetooth with applications registered and running on SDL and was also connected over USB, SDL must start reconnection timer and NOT send OnAppRegistered to HMI.

NOTE

SDL Apps that are using the websocket transport adapter will send `OnAppRegistered` after the user has activated the app and the websocket connection is opened. The HMI should not use `OnAppRegistered` for updating the available apps in the app list. BC.UpdateAppList should be used for updating the app list.

Notification

PARAMETERS

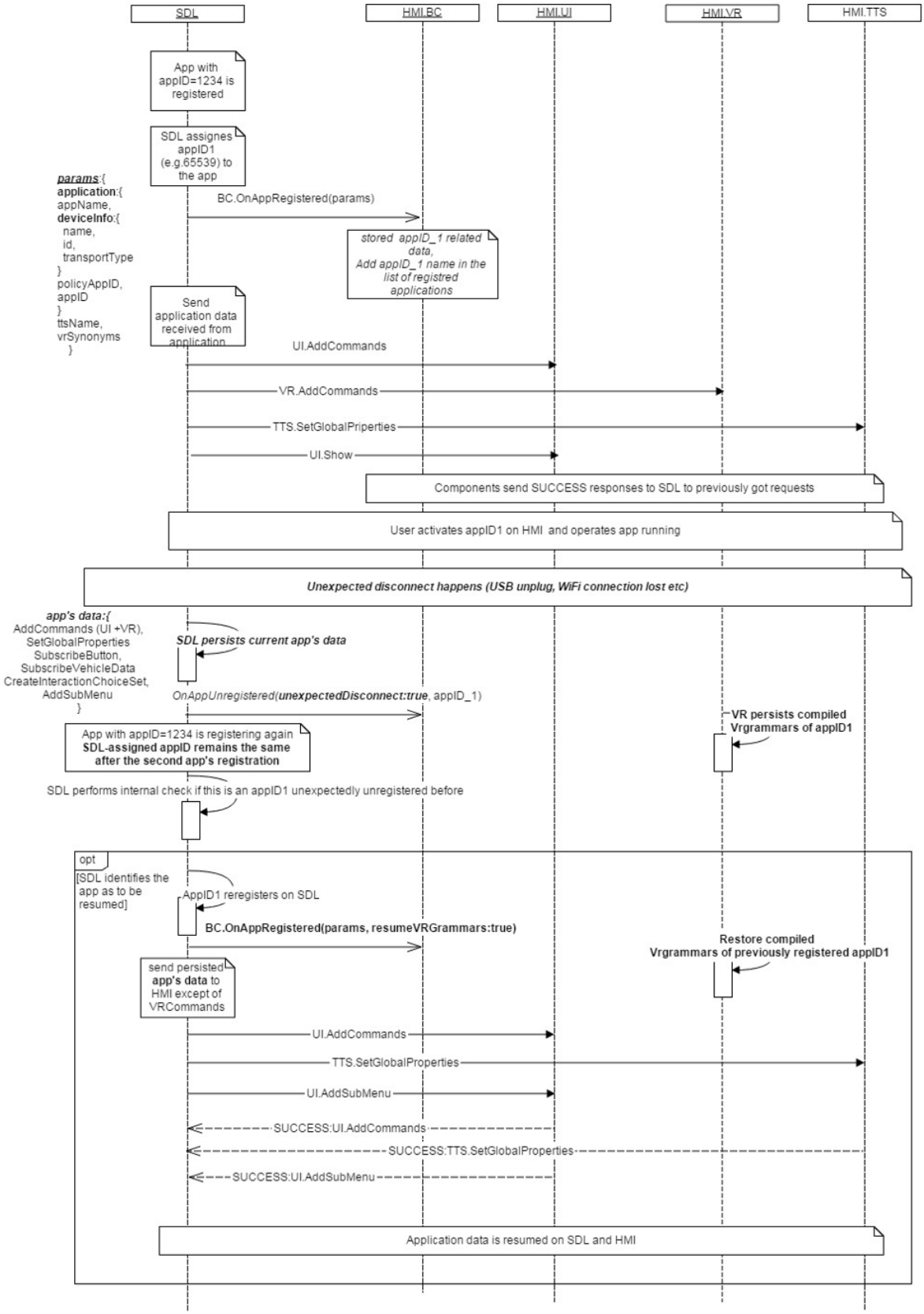
NAME	TYPE	MANDATORY	ADDITIONAL
application	Common.HMIApplication	true	
ttsName	Common.TTSChunk	false	array: true minsize: 1 maxsize: 100
vrSynonyms	String	false	array: true minsize: 1 maxsize: 100 maxlength: 40
resumeVrGrammars	Boolean	false	
priority	Common.AppPriority	false	

Sequence Diagrams

SEQUENCE DIAGRAM

App Register with Resume

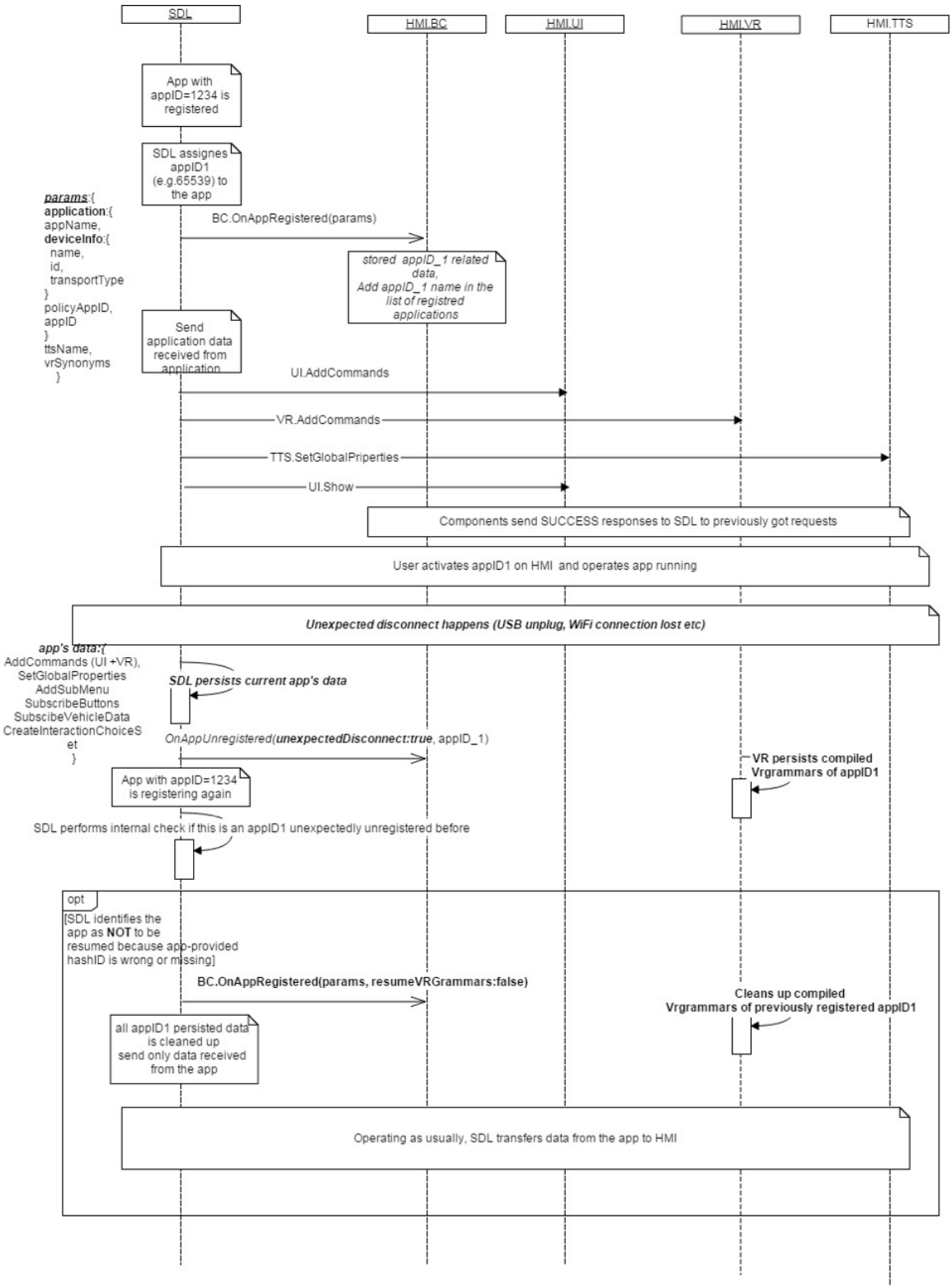
View Diagram



SEQUENCE DIAGRAM

App Register without Resume

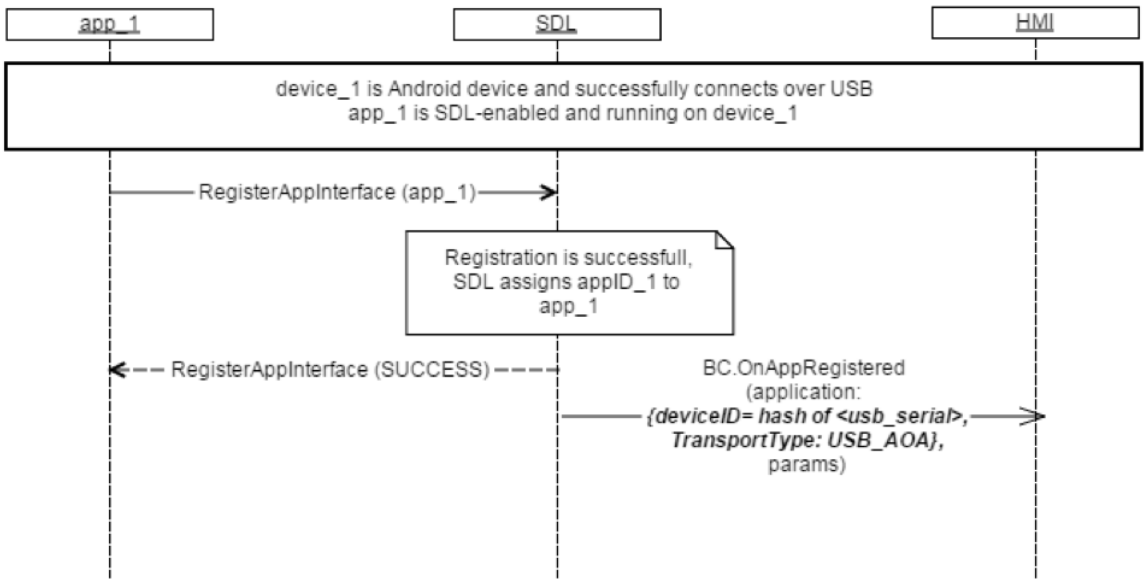
View Diagram



SEQUENCE DIAGRAM

App Registers on USB

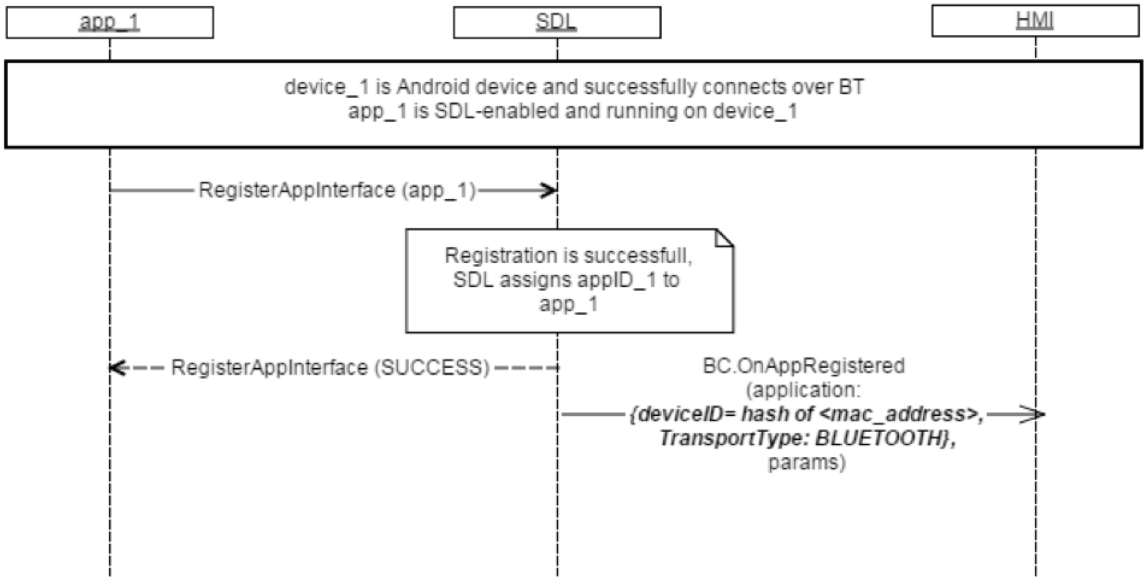
View Diagram



SEQUENCE DIAGRAM

App Registers on Bluetooth

View Diagram



JSON EXAMPLE NOTIFICATION

```
{
 "jsonrpc" : "2.0",
 "method" : "BasicCommunication.OnAppRegistered",
 "params" :
 {
  "application" :
   {
    "appName" : "TryMe",
    "ngnMediaScreenAppName " : "TryMe",
    "deviceInfo":
           { "name" : "GT-I9300",
          "id" : 1563462,
     "transportType" : "BLUETOOTH"
           },
           "policyAppID" : 123,
    "appID" : 65540,
    "hmiDisplayLanguageDesired" : ES-ES,
    "isMediaApplication" : false
   }
  "resumeVRGrammars" : true
 }
}
```

CLOUD APP JSON EXAMPLE NOTIFICATION

```
{
  "jsonrpc": "2.0",
  "method": "BasicCommunication.OnAppRegistered",
  "params": {
    "application": {
      "appID": 845030685,
      "appName": "SDL App",
      "appType": [
        "DEFAULT"
      ],
      "cloudConnectionStatus": "CONNECTED",
      "dayColorScheme": {
        "backgroundColor": {
          "blue": 98,
          "green": 78,
          "red": 36
        },
        "primaryColor": {
          "blue": 248,
          "green": 246,
          "red": 242
        },
        "secondaryColor": {
          "blue": 98,
          "green": 78,
          "red": 36
        }
      },
      "deviceInfo": {
        "id":
"1d92c1e33853c4566641d26e7043e7646da2a4d4a2fa0f99687fdf75f7f4
,
        "isSDLAllowed": true,
        "name": "ws://192.168.1.69:3000/",
        "transportType": "CLOUD_WEBSOCKET"
      },
      "hmiDisplayLanguageDesired": "EN-US",
      "isCloudApplication": true,
      "isMediaApplication": false,
      "ngnMediaScreenAppName": "SDLApp",
      "nightColorScheme": {
        "backgroundColor": {
          "blue": 35,
          "green": 17,
          "red": 2
        },
        "primaryColor": {
          "blue": 248,
          "green": 246,
          "red": 242
```

```
        },
        "secondaryColor": {
          "blue": 98,
          "green": 78,
          "red": 36
        }
      },
      "policyAppID": "123456",
      "requestSubType": [],
      "requestType": [],
      "vrSynonyms": [
        "SDLApp"
      ]
    },
    "priority": "NONE",
    "vrSynonyms": [
      "SDLApp"
    ]
  }
}
```


OnAppUnregistered

Type Notification
Sender SDL
Purpose Inform the HMI that an application has unregistered.

SDL sends this notification when:

1. The named application has unregistered from the mobile side via an appropriate RPC.
2. The connection and corresponding session(s) are closed due to transport connection issues (for example, WiFi/BT connection closing on mobile device, USB unplugging, etc.).
3. A HeartBeat timeout occurs between the mobile device and SDL Core (if a HeartBeat timeout is set).

MUST

1. HMI must distinguish distinguish between an unexpected disconnect and a regular exit from the `unexpectedDisconnect` parameter.
2. HMI must removed the named application from the list of registered applications.

MAY

1. In case of an `Unexpected Disconnect`, the HMI may display a popup of some kind that reports a connection with the named app was unexpectedly lost.
2. In case of any type of disconnect, the HMI may switch to a `Home` screen that displays a list of other registered applications.
3. The HMI may delete all application data after an app disconnects, except for `VRGrammars`.

NOTE

Information about the application (name, appID, etc) is provided by SDL via BC.UpdateAppList or BC.OnAppRegistered.

Notification

PARAMETERS

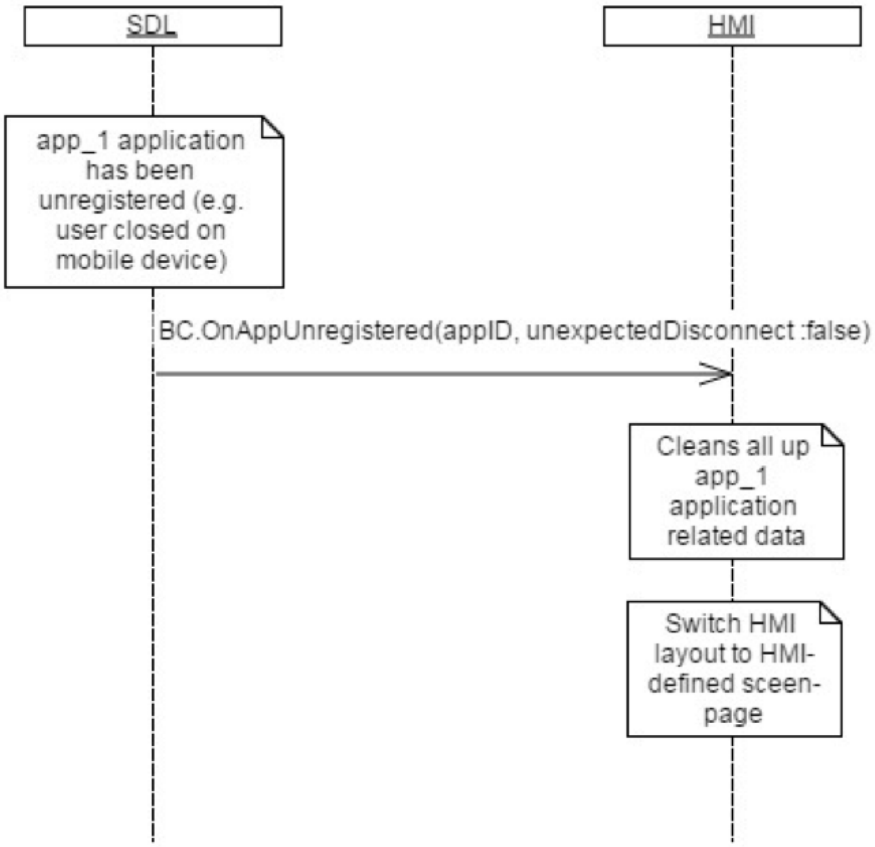
NAME	TYPE	MANDATORY	ADDITIONAL
unexpectedDisconnect	Boolean	true	
appID	Integer	true	

Sequence Diagrams

SEQUENCE DIAGRAM

Active App Unregistered

View Diagram



JSON EXAMPLE NOTIFICATION

```
{
 "jsonrpc" : "2.0",
 "method" : "BasicCommunication.OnAppUnregistered",
 "params" :
 {
  "appID" : 65539,
   "unexpectedDisconnect":"false"
 }
}
```

OnAwakeSDL

Type Notification
Sender HMI
Purpose Notify SDL to return to normal operation after a suspend event.

This notification is sent to SDL to notify that there won't be `IGNITION_OFF` sent after OnExitAllApplications(`SUSPEND`), which was previously sent by the HMI.

MAY

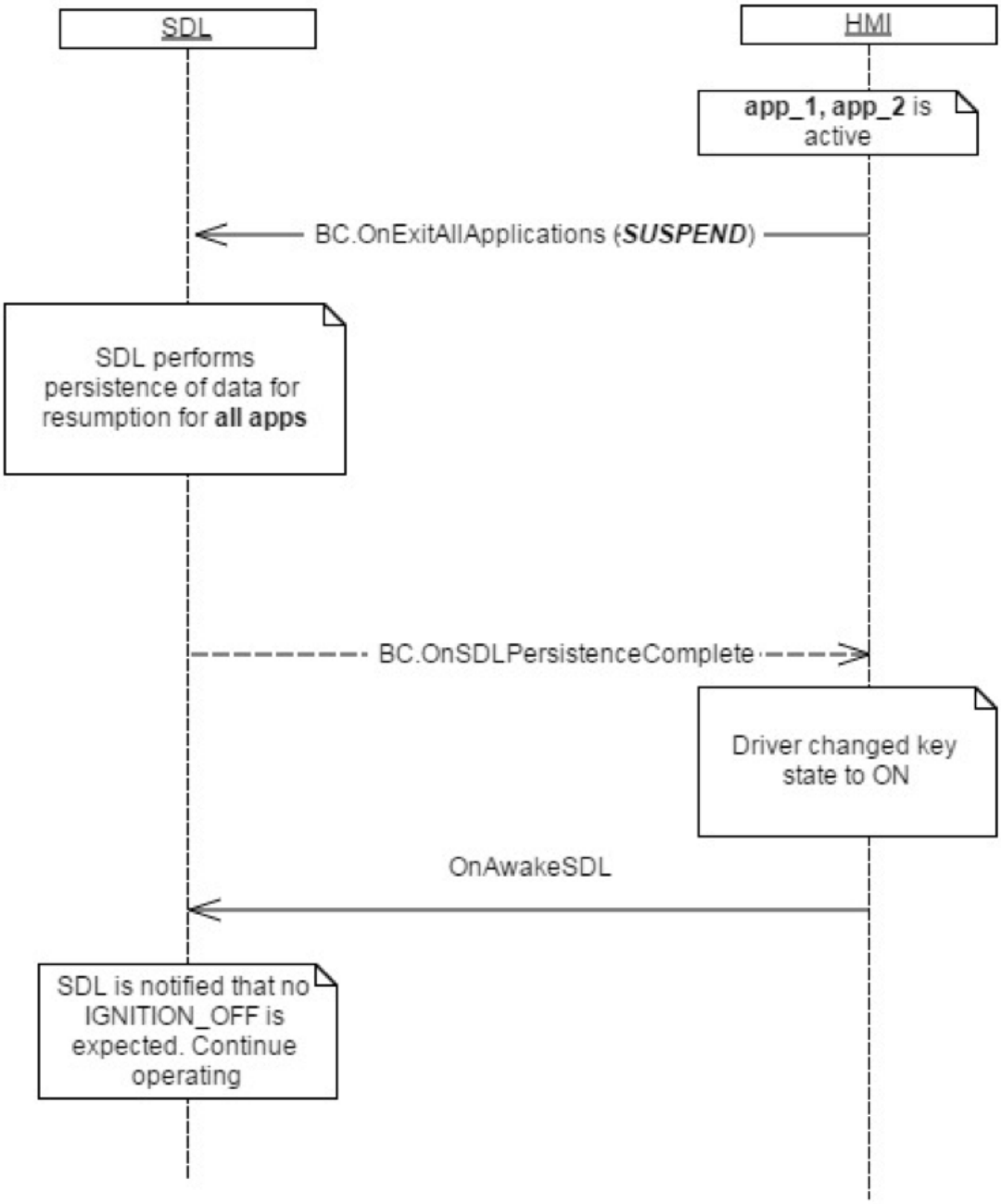
1. Send `OnAwakeSDL` if the `ACC` does not change, but the driver door is opened and closed two times.
2. Send `OnAwakeSDL` ifif the `ACC` key position is set to `On`.

Sequence Diagrams

SEQUENCE DIAGRAM

Wake SDL from Suspension

View Diagram



JSON EXAMPLE NOTIFICATION

```
{
  "jsonrpc" : "2.0",
  "method" : "BasicCommunication. OnAwakeSDL"
}
```

OnExitAllApplications

Type Notification
Sender HMI
Purpose Inform SDL to exit every registered application.

SDL requires this notification in order to accurately close the sessions with registered applications before reloading or shutting down based on the user's actions.

MUST

- Send `OnExitAllApplications` with the appropriate `reason` upon one of the users's actions:
  - Master Reset
  - Key set to Ignition Off (Refer to the diagram below).
  - Key set to Suspend (Refer to the diagram below).
  - Key set to ACC

Notification

PARAMETERS

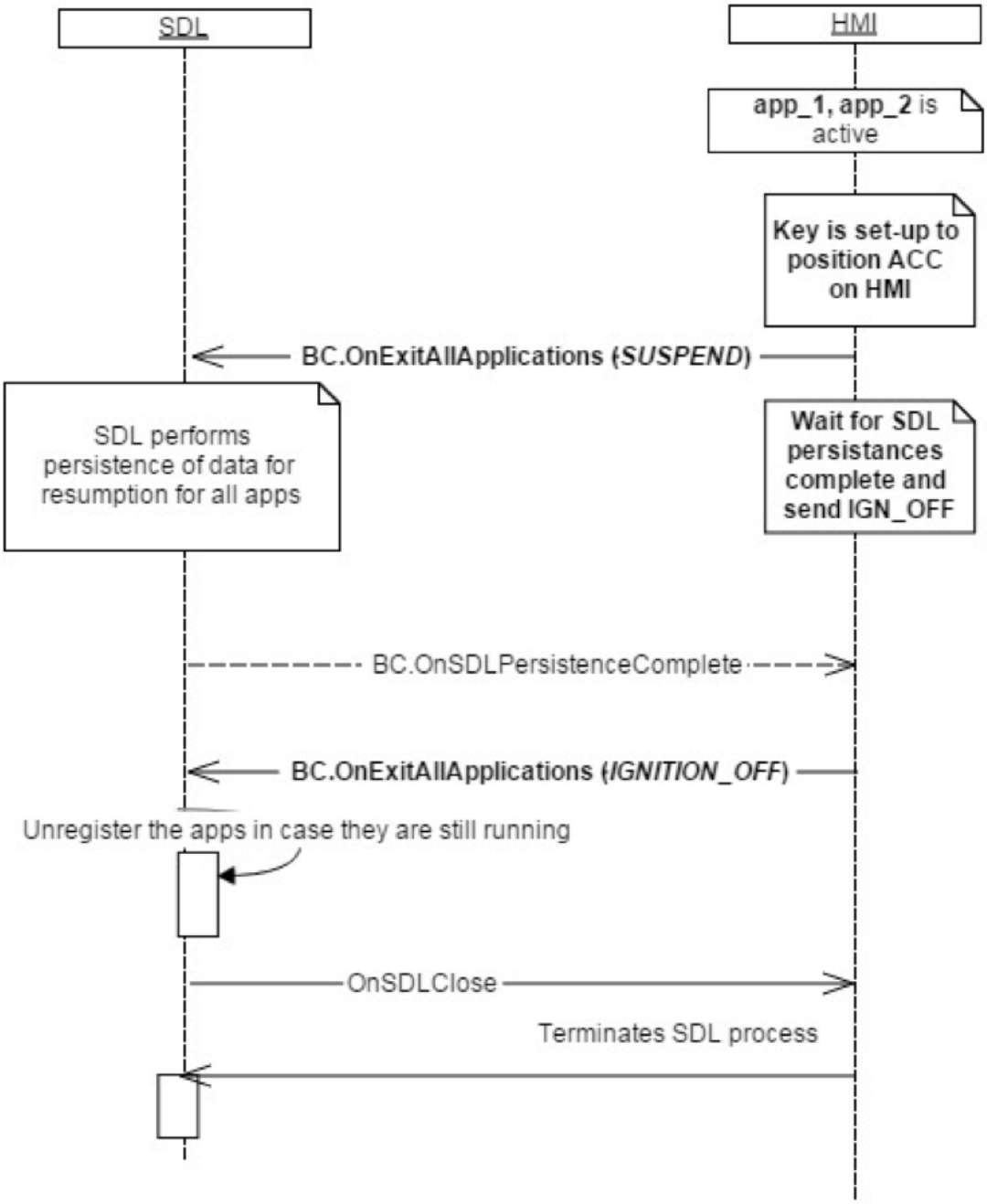
NAME	TYPE	MANDATORY	ADDITIONAL
reason	Common.ApplicationsCloseReason	true	

Sequence Diagrams

SEQUENCE DIAGRAM

Exit All Apps on Ignition Off

View Diagram


SDL
HMI
app_1, app_2 is active
Key is set-up to position ACC on HMI
BC.OnExitAllApplications (SUSPEND)
SDL performs persistence of data for resumption for all apps
Wait for SDL persistances complete and send IGN_OFF
BC.OnSDLPersistenceComplete
BC.OnExitAllApplications (IGNITION_OFF)
Unregister the apps in case they are still running
OnSDLClose
Terminates SDL process

JSON EXAMPLE NOTIFICATION

```
{
 "jsonrpc" : "2.0",
 "method" : "BasicCommunication.OnExitAllApplications",
 "params" :
 {
  "reason" : "IGNITION_OFF"
 }
}
```

OnExitApplication

Type Notification
Sender HMI
Purpose Inform SDL that an application must be unregistered or put into the NONE HMI state.

MUST

1. Provide the possibility for the user to exit any of the registered applications.
2. Track if an application is active or running in the background, and complies with the driver distraction rules of the system.
3. Track that the transport type that corresponds to the application running. For example, navigation applications are not allowed to be run over a bluetooth connection.
4. Ignore all SDL RPC's related to an application that are defined as prohibited for a specific transport type.
5. Send `OnExitApplication` notification to SDL if an application conflicts with any of the rules above (i.e. Users requests to exit, driver distraction rules violated, or navigation app connected over bluetooth).

MAY

The HMI may switch layouts or views according to the workflow after deactivation of an application.

NOTE

- Information about the application (name, appID, etc) is provided by SDL via BC.UpdateAppList or BC.OnAppRegistered.
- SDL ignores all invalid notifications which come from the HMI (Invalid JSON, invalid data types/bounds, etc).
- If the HMI sends ApplicationExitReason `CLOSE_CLOUD_CONNECTION`
  - The application will be unregistered.
  - A BasicCommunication.UpdateDeviceList (with the closed application) will be sent by SDL.
  - The closed application will have a cloudConnectionStatus of `NOT_CONNECTED`

Notification

PARAMETERS

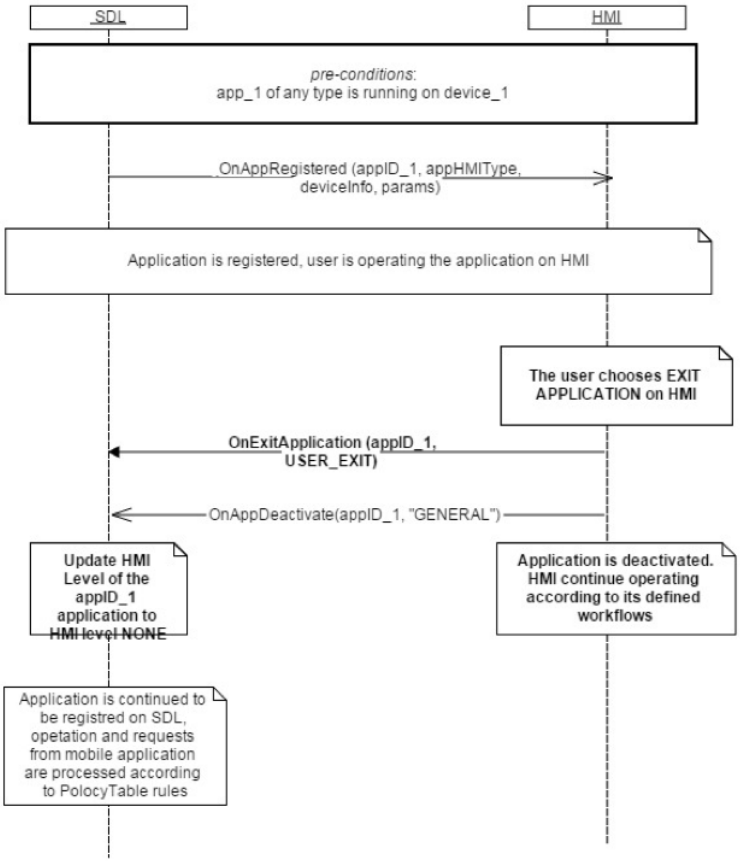
NAME	TYPE	MANDATORY	ADDITIONAL
reason	Common.ApplicationExitReason	true	
appID	Integer	true	

Sequence Diagrams

SEQUENCE DIAGRAM

User Exits Application

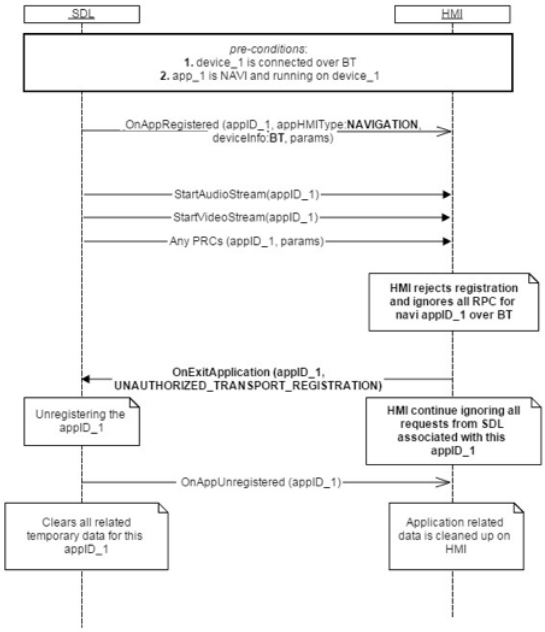
View Diagram



SEQUENCE DIAGRAM

Application Not Authorized

View Diagram



JSON EXAMPLE NOTIFICATION

```
{
 "jsonrpc" : "2.0",
 "method" : "BasicCommunication.OnExitApplication",
 "params" :
 {
  "appID" : 65544,
  "reason" : "USER_EXIT"
 }
}
```

OnEventChanged

Type Notification
Sender HMI
Purpose Notify SDL that an event becomes active or inactive

SDL uses `OnEventChanged` notification to appropriately manage the hmiLevel and audioStreamingState of each application during an active event.

PHONE_CALL

MUST

1. Send notification with appropriate parameter value when active call on HMI has been started/ended.
2. Send SDL.OnAppDeactivated to the active app when the phone call is started.
3. Resume the applications to their original state prior to the phone call event in the HMI when the event ends (see note below).

NOTE

SDL does not send BC.ActivateApp or BC.OnResumeAudioSource to HMI after the phone call is ended.

Upon receiving `OnEventChanged(PHONE_CALL)`, SDL will:

ISACTIVE	RESULT
true	Change the HMI state of all applications currently either in FULL or LIMITED to (BACKGROUND, NOT_AUDIBLE)
false	Return applications to the same HMI state they had prior to the event

EMERGENCY_EVENT

EMERGENCY_EVENT is an HMI-specific event used when "Emergency event" or "Rear view camera" are active. The main idea of this from the SDL<->HMI point of view is that navigation/audio streaming mustn't interfere with Rear Camera View mode. The HMI is responsible for managing audio/video data while EMERGENCY_EVENT is active.

MUST

1. Send a notification with the appropriate parameter value when EMERGENCY_EVENT becomes active or inactive.

Upon receiving `OnEventChanged(EMERGENCY_EVENT)`, SDL will:

ISACTIVE	RESULT
true	Move all apps with audioStreamingState AUDIBLE to NOT_AUDIBLE
false	Return applications to the same HMI state they had prior to the event

> NOTE
>
> While the event is active, the app is not allowed to stream audio and it will not be heard by the user (due to other audio and/or system events blocking it).

DEACTIVATE_HMI

> MUST
>
> 1. Send notification with appropriate parameter value when all apps should be deactivated/restored.
> 2. Send `OnEventChanged(DEACTIVATE_HMI, isActive: false)` before activating an app.

Upon receiving `OnEventChanged(DEACTIVATE_HMI)`, SDL will:

ISACTIVE	RESULT
true	Change the hmiLevel of all applications currently in (FULL/LIMITED) to (BACKGROUND, NOT_AUDIBLE)
false	Return applications to the same HMI state they had prior to the event

NOTE

When this event is active, SDL **rejects** all app activation requests from the HMI.

AUDIO_SOURCE/EMBEDDED_NAVI

MUST

1. Send notification to SDL with appropriate parameter value when embedded navigation or audio source is activated/deactivated.
2. Send `SDL.ActivateApp(appID)` in case of app activation or `BC.OnAppDeactivated(appID)` in case of app deactivation.
3. Switch off embedded source before app activation, when the type of activating app and embedded source are the same:
    - The HMI must deactivate the AUDIO_SOURCE event if a media app is activated.
    - The HMI must deactivate the EMBEDDED_NAVI event if a navigation app is activated.
4. When the system supports audio mixing and embedded navigation starts streaming
    - Send TTS.Started to SDL to change media app currently in (LIMITED, AUDIBLE) to (LIMITED, ATTENUATED) due to active embedded navigation.

- Send TTS.Stopped to SDL right after embedded navigation stops streaming to change application's HMIStatus to the same state it had prior to the event.

NOTE

- When app is successfully registered and SDL receives `OnEventChanged(AUDIO_SOURCE, isActive:true)` or `OnEventChanged(EMBEDDED_NAVI, isActive:true)`, SDL changes hmiLevel and audioStreamingState of this application.
  - See the table *HMI Status of apps when AUDIO_SOURCE or EMBEDDED_NAVI event is activated*
- When app is activated during an active EMBEDDED_NAVI or AUDIO_SOURCE event, SDL sets the appropriate hmiLevel and audioStreamingState for the app.
  - See the table *Activating apps during active AUDIO_SOURCE or EMBEDDED_NAVI event*
- Given that a system supports audio mixing ("MixingAudioSupported" = true at .ini file), then:
  - If there is a navigation app in (FULL/LIMITED, AUDIBLE) and SDL receives `OnEventChanged(AUDIO_SOURCE, isActive=true)`, then SDL will change the navigation app's state to (LIMITED, AUDIBLE)
  - If there is a navigation app that is in (LIMITED, AUDIBLE) due to an active AUDIO_SOURCE event, and SDL receives `SDL.ActivateApp(appID_of_navigation_app)`, then SDL will change the navigation app's state to (FULL, AUDIBLE).
  - If SDL receives `OnEventChanged(EMBEDDED_NAVI, isActive=true)`, SDL changes any media app in (LIMITED, AUDIBLE) to (LIMITED, ATTENUATED). After the EMBEDDED_NAVI event ends, SDL changes the media app's state to (LIMITED, AUDIBLE).

HMI STATUS OF APPS WHEN `AUDIO_SOURCE` OR `EMBEDDED_NAVI` EVENT IS ACTIVATED

APPHMITYPE	EVENT	HMI STATE BEFORE	HMI STATE AFTER
Media	AUDIO_SOURCE	(FULL/LIMITED, AUDIBLE)	(BACKGROUND, NOT_AUDIBLE)
Navigation	AUDIO_SOURCE	(FULL/LIMITED, AUDIBLE)	(LIMITED, AUDIBLE)
Non-media	AUDIO_SOURCE	(FULL/LIMITED, AUDIBLE)	(BACKGROUND, NOT_AUDIBLE)
Media	EMBEDDED_NAVI	(FULL/LIMITED, AUDIBLE)	(LIMITED, AUDIBLE)
Navigation	EMBEDDED_NAVI	(FULL/LIMITED, AUDIBLE)	(BACKGROUND, NOT_AUDIBLE)
Non-media	EMBEDDED_NAVI	(FULL/LIMITED, AUDIBLE)	(BACKGROUND, NOT_AUDIBLE)

ACTIVATING APPS DURING ACTIVE `AUDIO_SOURCE` OR `EMBEDDED_NAVI` EVENT

APPHMITYPE	EVENT	NEW HMI STATE	KEEP EVENT ACTIVE
Media	AUDIO_SOURCE	(FULL, AUDIBLE)	false
Navigation	AUDIO_SOURCE	(FULL, AUDIBLE)	true
Non-media	AUDIO_SOURCE	(FULL, NOT_AUDIBLE)	true
Media	EMBEDDED_NAVI	(FULL, AUDIBLE)	true
Navigation	EMBEDDED_NAVI	(FULL, AUDIBLE)	false
Non-media	EMBEDDED_NAVI	(FULL, NOT_AUDIBLE)	true

PARAMETERS

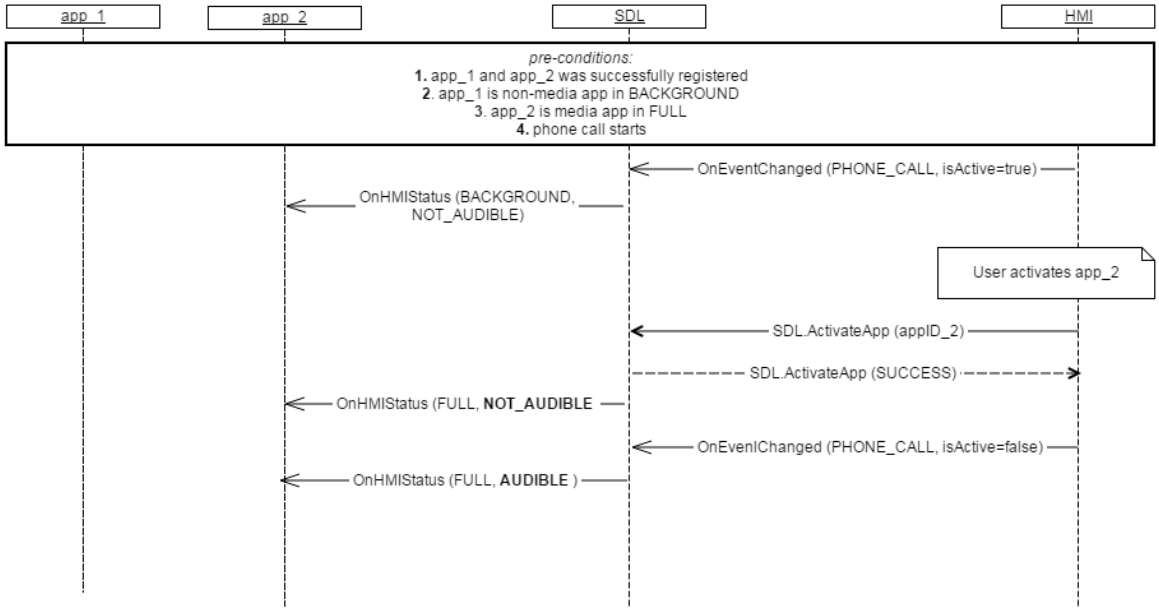
NAME	TYPE	MANDATORY	ADDITIONAL	DESCRIPTION
eventName	Common.EventTypes	true	-	Specifies the types of active events Must be 'true' when the event is started on HMI.
isActive	Boolean	true	-	Must be 'false' when the event is ended on HMI

Sequence Diagrams

SEQUENCE DIAGRAM

PHONE_CALL, media app is active

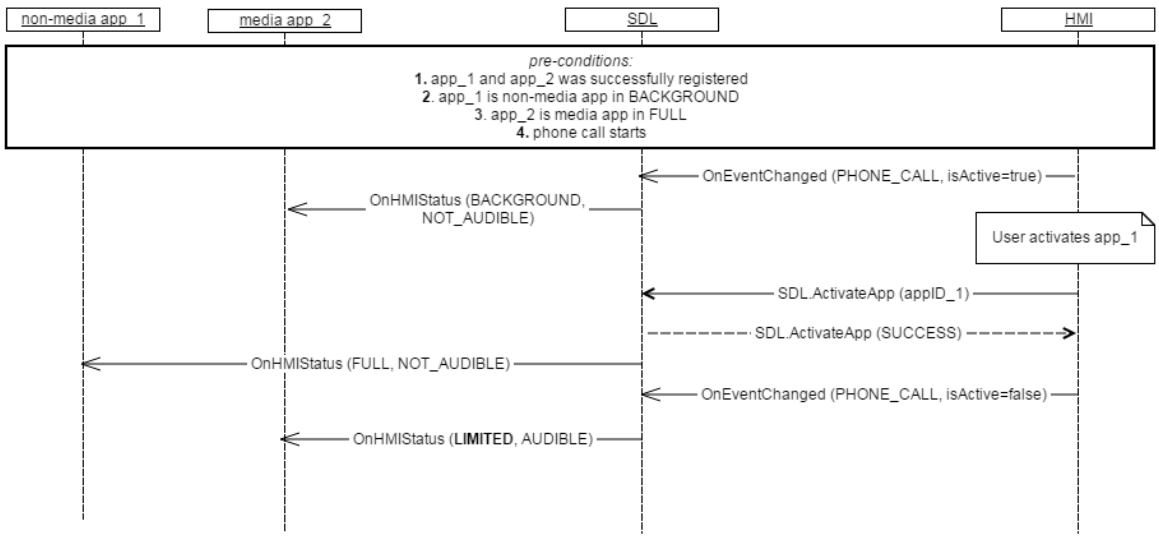
View Diagram



SEQUENCE DIAGRAM

PHONE_CALL, non-media app is active

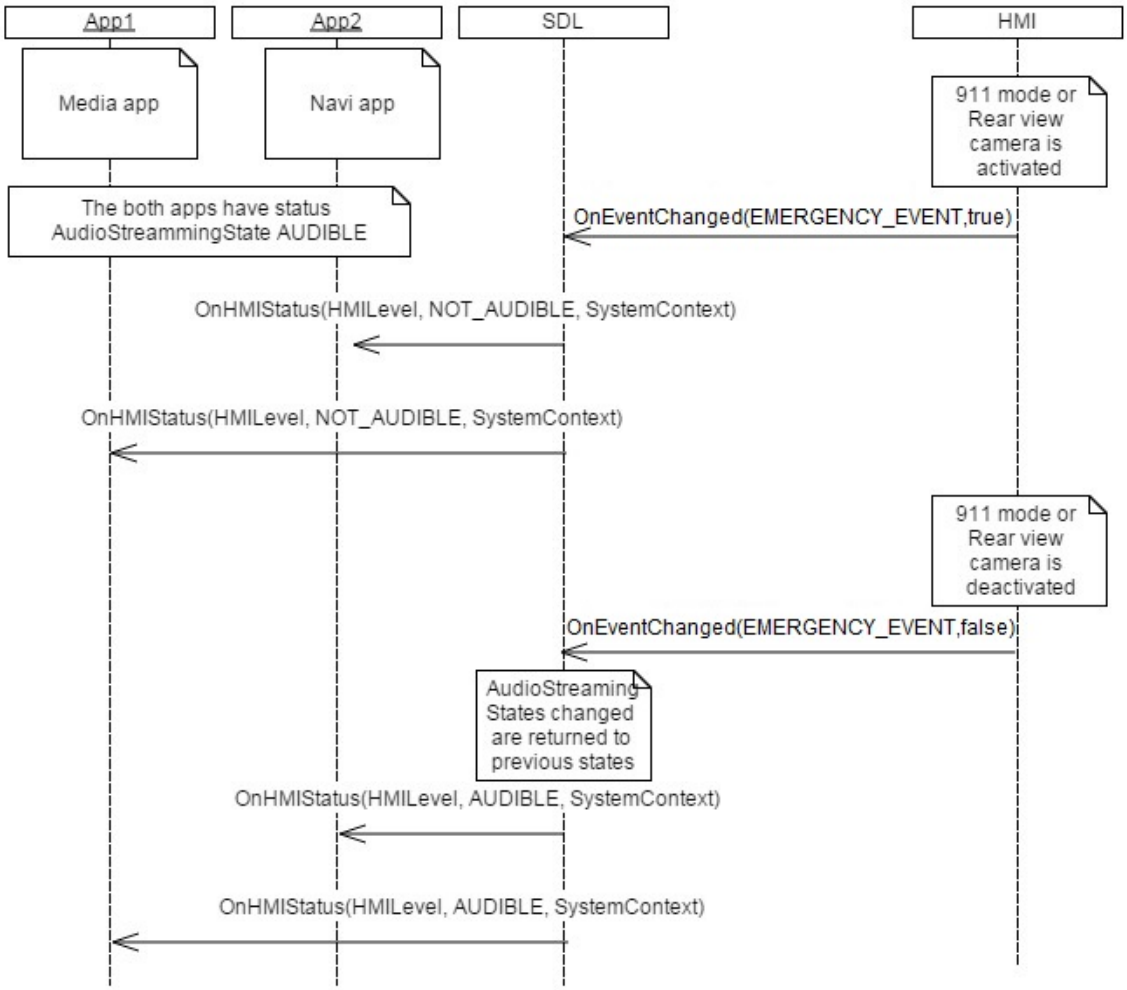
View Diagram



SEQUENCE DIAGRAM

EMERGENCY_EVENT

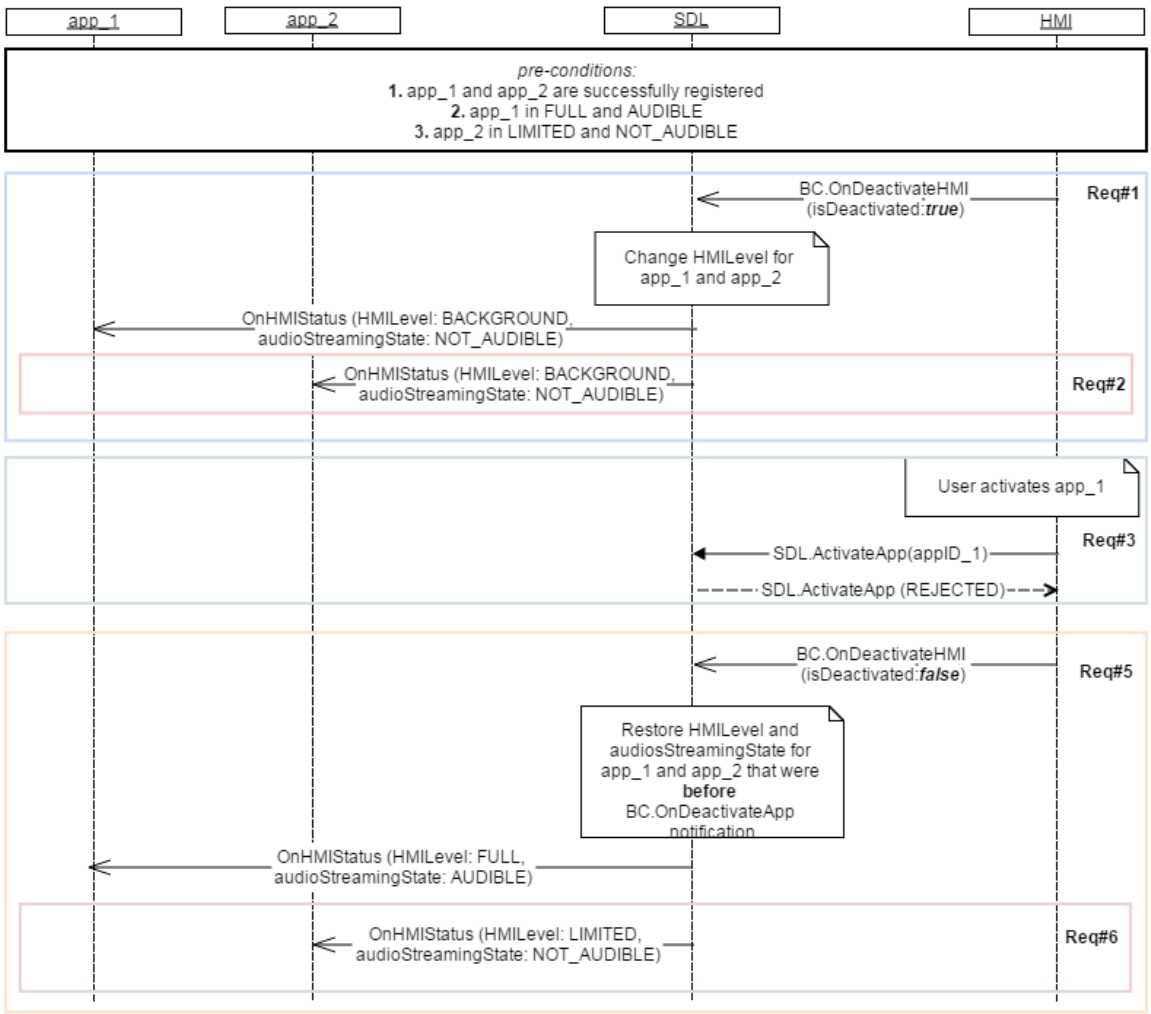
View Diagram



SEQUENCE DIAGRAM

DEACTIVATE_HMI

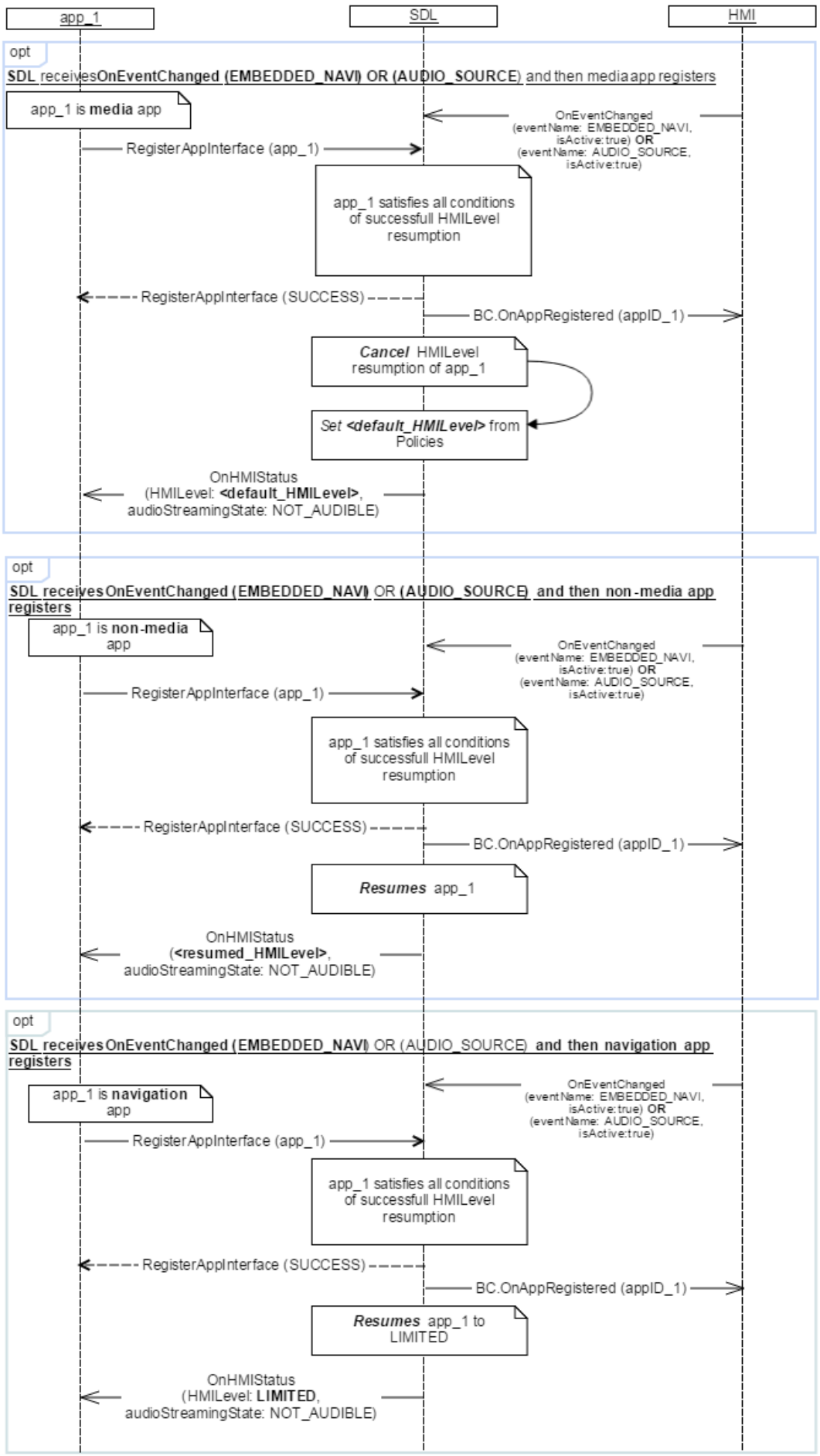
View Diagram



SEQUENCE DIAGRAM

EMBEDDED_NAVI or AUDIO_SOURCE

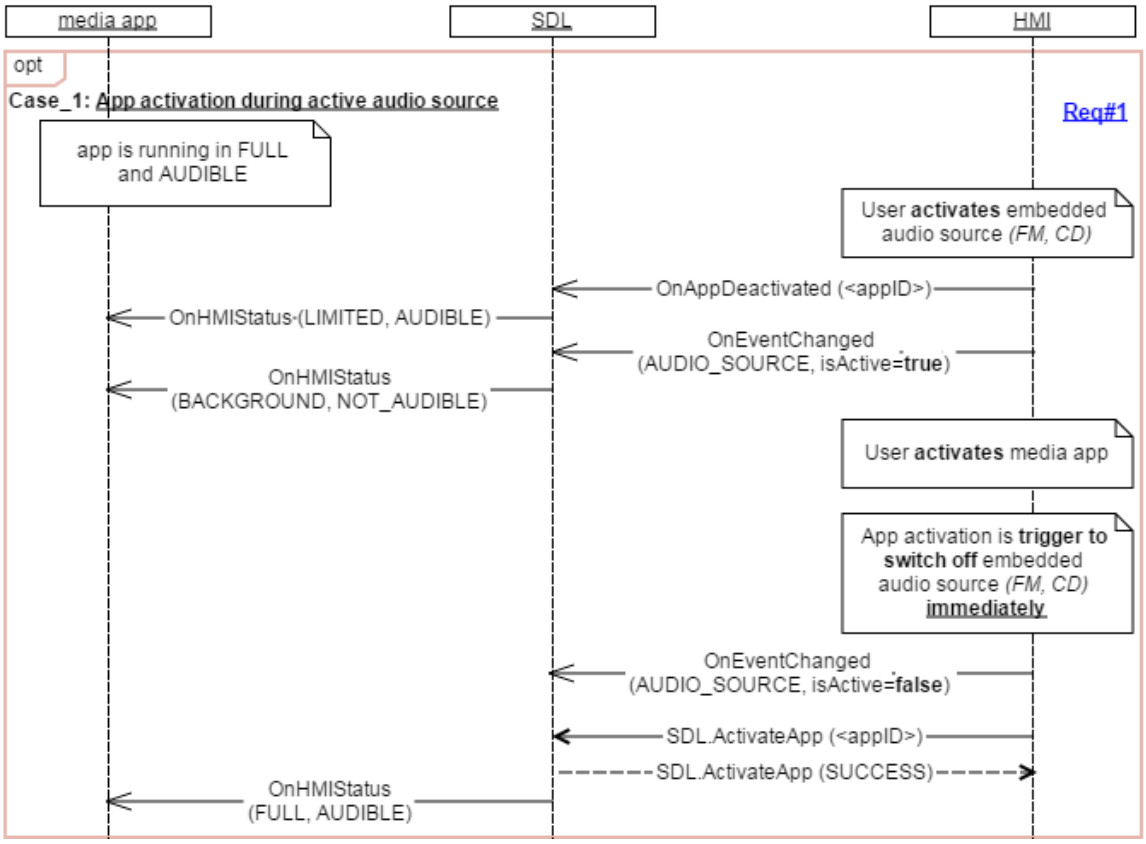
View Diagram


app_1
SDL
HMI
opt
SDL receives OnEventChanged (EMBEDDED_NAVI) OR (AUDIO_SOURCE) and then media app registers
app_1 is media app
OnEventChanged (eventName: EMBEDDED_NAVI, isActive: true) OR (eventName: AUDIO_SOURCE, isActive: true)
RegisterAppInterface (app_1)
app_1 satisfies all conditions of successfull HMILevel resumption
RegisterAppInterface (SUCCESS)
BC.OnAppRegistered (appID_1)
Cancel HMILevel resumption of app_1
Set <default_HMILevel> from Policies
OnHMIStatus (HMILevel: <default_HMILevel>, audioStreamingState: NOT_AUDIBLE)
opt
SDL receives OnEventChanged (EMBEDDED_NAVI) OR (AUDIO_SOURCE) and then non-media app registers
app_1 is non-media app
OnEventChanged (eventName: EMBEDDED_NAVI, isActive: true) OR (eventName: AUDIO_SOURCE, isActive: true)
RegisterAppInterface (app_1)
app_1 satisfies all conditions of successfull HMILevel resumption
RegisterAppInterface (SUCCESS)
BC.OnAppRegistered (appID_1)
Resumes app_1
OnHMIStatus (<resumed_HMILevel>, audioStreamingState: NOT_AUDIBLE)
opt
SDL receives OnEventChanged (EMBEDDED_NAVI) OR (AUDIO_SOURCE) and then navigation app registers
app_1 is navigation app
OnEventChanged (eventName: EMBEDDED_NAVI, isActive: true) OR (eventName: AUDIO_SOURCE, isActive: true)
RegisterAppInterface (app_1)
app_1 satisfies all conditions of successfull HMILevel resumption
RegisterAppInterface (SUCCESS)
BC.OnAppRegistered (appID_1)
Resumes app_1 to LIMITED
OnHMIStatus (HMILevel: LIMITED, audioStreamingState: NOT_AUDIBLE)

SEQUENCE DIAGRAM

App activation during active audio source

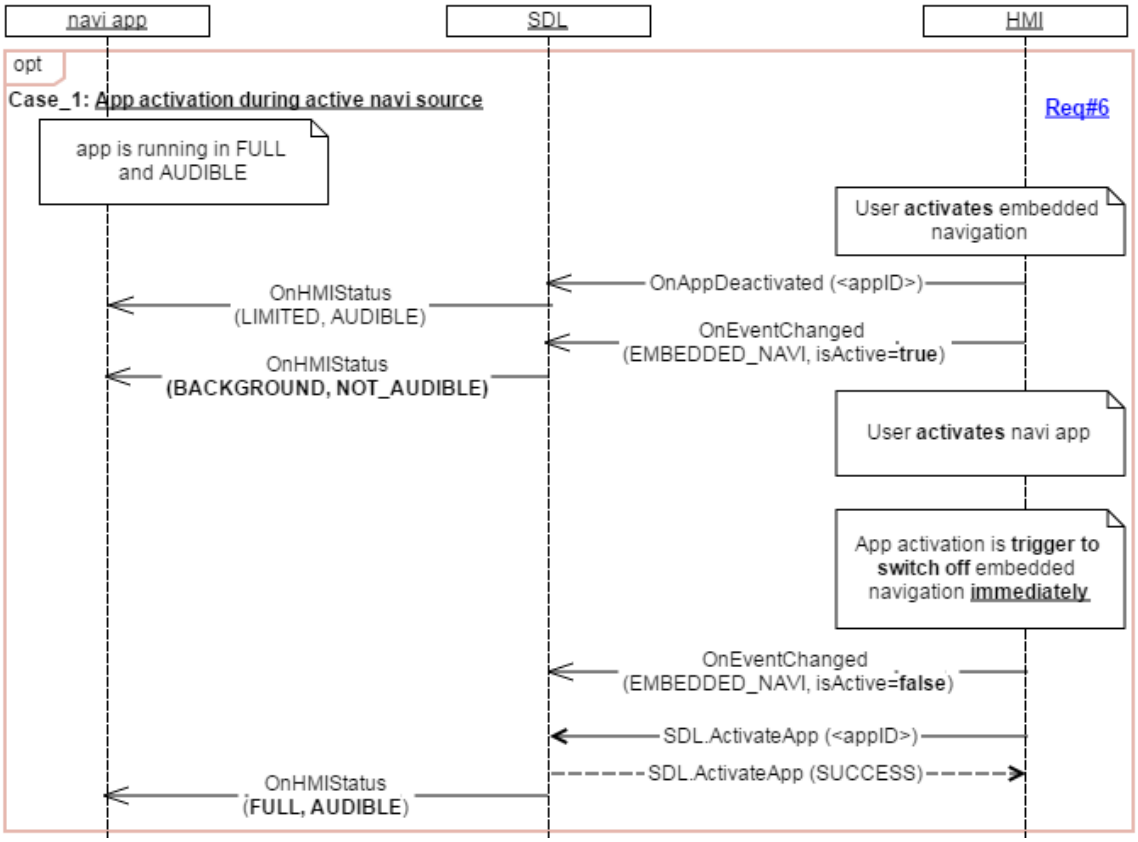
View Diagram



SEQUENCE DIAGRAM

App activation during active embedded navigation

View Diagram

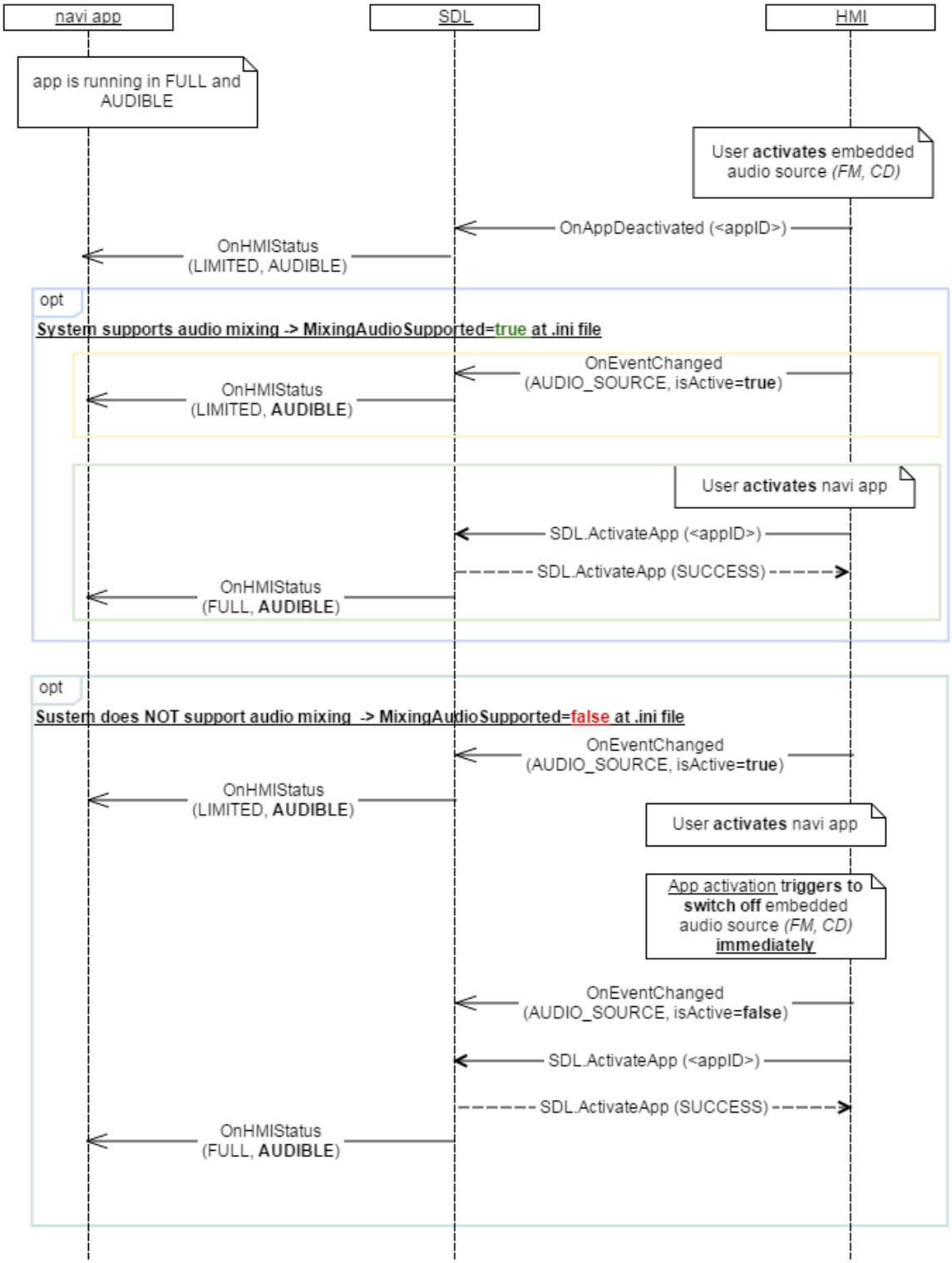


SEQUENCE DIAGRAM

Correlation between audioStreamingState of media app and embedded navigation in case "MixingAudioSupported" = true/false

---

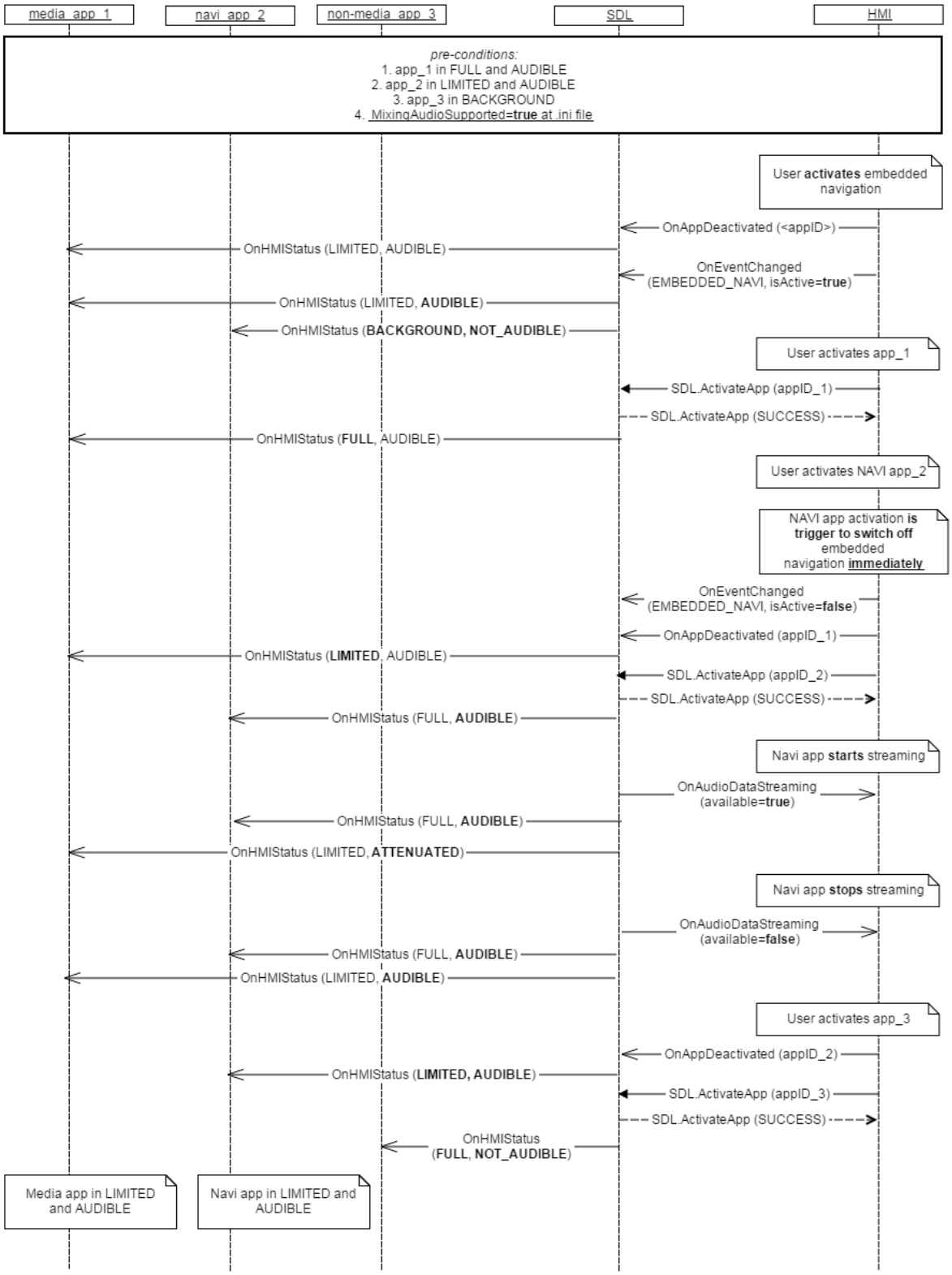
View Diagram



SEQUENCE DIAGRAM

Multiple apps activation during active embedded navigation or audio source

View Diagram


media_app_1
navi_app_2
non-media_app_3
SDL
HMI
pre-conditions:
1. app_1 in FULL and AUDIBLE
2. app_2 in LIMITED and AUDIBLE
3. app_3 in BACKGROUND
4. MixingAudioSupported=true at .ini file
User activates embedded navigation
OnAppDeactivated (<appID>)
OnHMIStatus (LIMITED, AUDIBLE)
OnEventChanged (EMBEDDED_NAVI, isActive=true)
OnHMIStatus (LIMITED, AUDIBLE)
OnHMIStatus (BACKGROUND, NOT_AUDIBLE)
User activates app_1
SDL.ActivateApp (appID_1)
SDL.ActivateApp (SUCCESS)
OnHMIStatus (FULL, AUDIBLE)
User activates NAVI app_2
NAVI app activation is trigger to switch off embedded navigation immediately
OnEventChanged (EMBEDDED_NAVI, isActive=false)
OnAppDeactivated (appID_1)
OnHMIStatus (LIMITED, AUDIBLE)
SDL.ActivateApp (appID_2)
SDL.ActivateApp (SUCCESS)
OnHMIStatus (FULL, AUDIBLE)
Navi app starts streaming
OnAudioDataStreaming (available=true)
OnHMIStatus (FULL, AUDIBLE)
OnHMIStatus (LIMITED, ATTENUATED)
Navi app stops streaming
OnAudioDataStreaming (available=false)
OnHMIStatus (FULL, AUDIBLE)
OnHMIStatus (LIMITED, AUDIBLE)
User activates app_3
OnAppDeactivated (appID_2)
OnHMIStatus (LIMITED, AUDIBLE)
SDL.ActivateApp (appID_3)
SDL.ActivateApp (SUCCESS)
OnHMIStatus (FULL, NOT_AUDIBLE)
Media app in LIMITED and AUDIBLE
Navi app in LIMITED and AUDIBLE

JSON EXAMPLE NOTIFICATION

```
{
 "jsonrpc" : "2.0",
 "method" : "OnEventChanged",
 "params" :
 {
  "eventName" : "PHONE_CALL",
  "isActive" : true
 }

}
```

OnFileRemoved

Type Notification
Sender SDL
Purpose Inform the HMI that a file has been removed from a shared folder by an application.

When a file is requested to be removed by an application, SDL will notify the HMI that the file has been removed.

MAY

If the named file that was removed was an app icon, the HMI may want to reset the app icon it was displaying to a default app icon.

NOTE

SDL notifies the HMI about removing files only if the files were located in the HMI's shared folder.

Notification

PARAMETERS

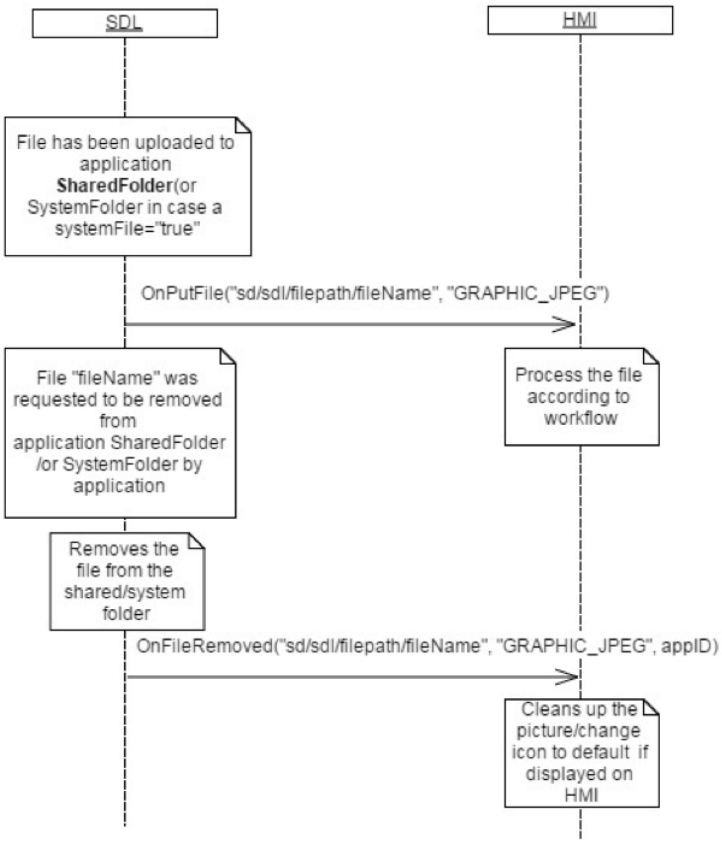
NAME	TYPE	MANDATORY	ADDITIONAL
fileName	String	true	minlength: 1 maxlength: 30
fileType	Common.FileType	true	
appID	Integer	true	

Sequence Diagrams

SEQUENCE DIAGRAM

File Removed from Head Unit

View Diagram



JSON EXAMPLE NOTIFICATION

```
{
  "jsonrpc" : "2.0",
  "method" : "BasicCommunication.OnFileRemoved"
     "params" :
  {
     "appID" : 31780,
"fileName" : "/fs/rwdata/storage/sdl/Emergency584421907/
syncFileName",
     "fileType" : "GRAPHIC_BMP"
  }

}
```

OnFindApplications

Type Notification
Sender HMI
Purpose Initiate the search for applications on the named device.

Notification

On receipt of this notification, SDL will provide the list of registered applications for the named device via the 'UpdateAppList' RPC.

NOTE

SDL ignores all invalid notifications sent from the HMI (e.g. invalid JSON, invalid data types).

MUST

HMI must send an 'OnFindApplications' notification when the user requests to find applications on a device (i.e. user chooses a device from a list of devices).

PARAMETERS

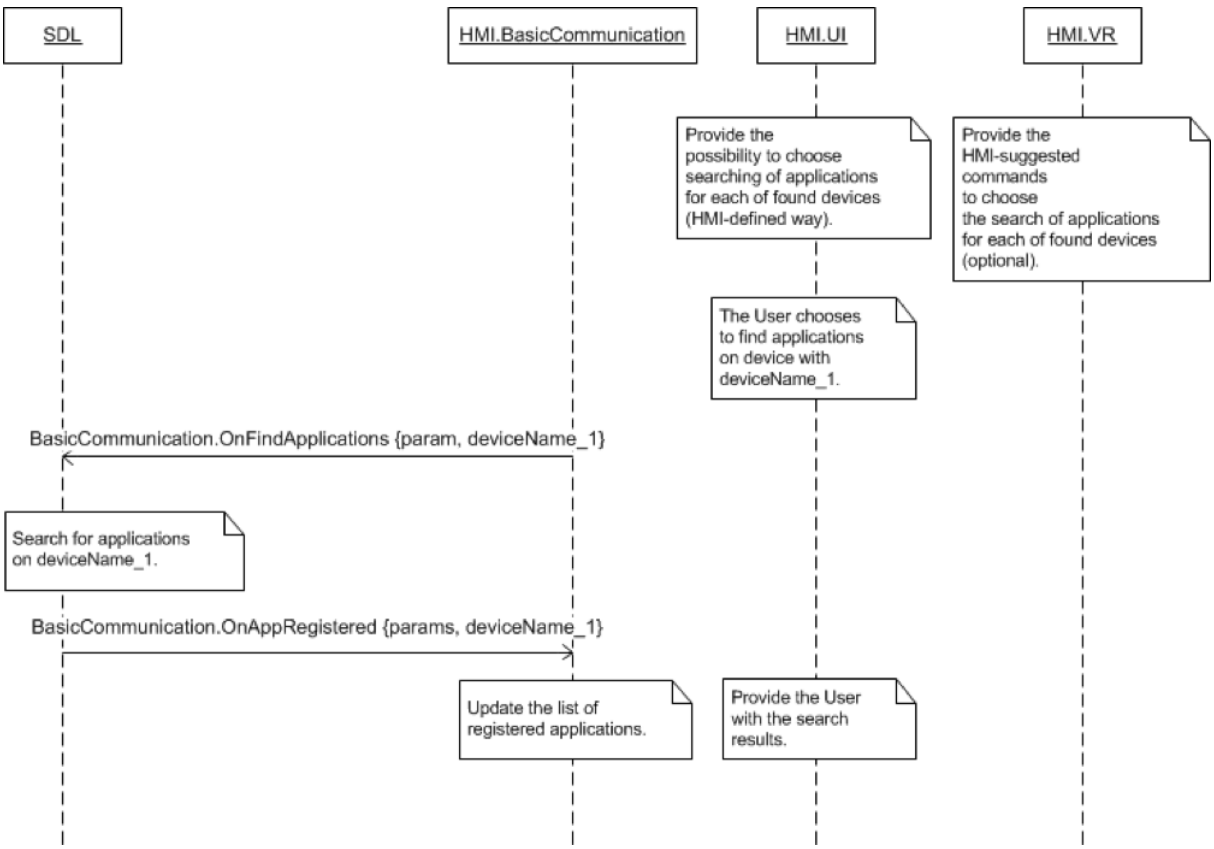
NAME	TYPE	MANDATORY	ADDITIONAL
deviceInfo	Common.DeviceInfo	false	

Sequence Diagrams

SEQUENCE DIAGRAM

User Choice After OnAppRegistered

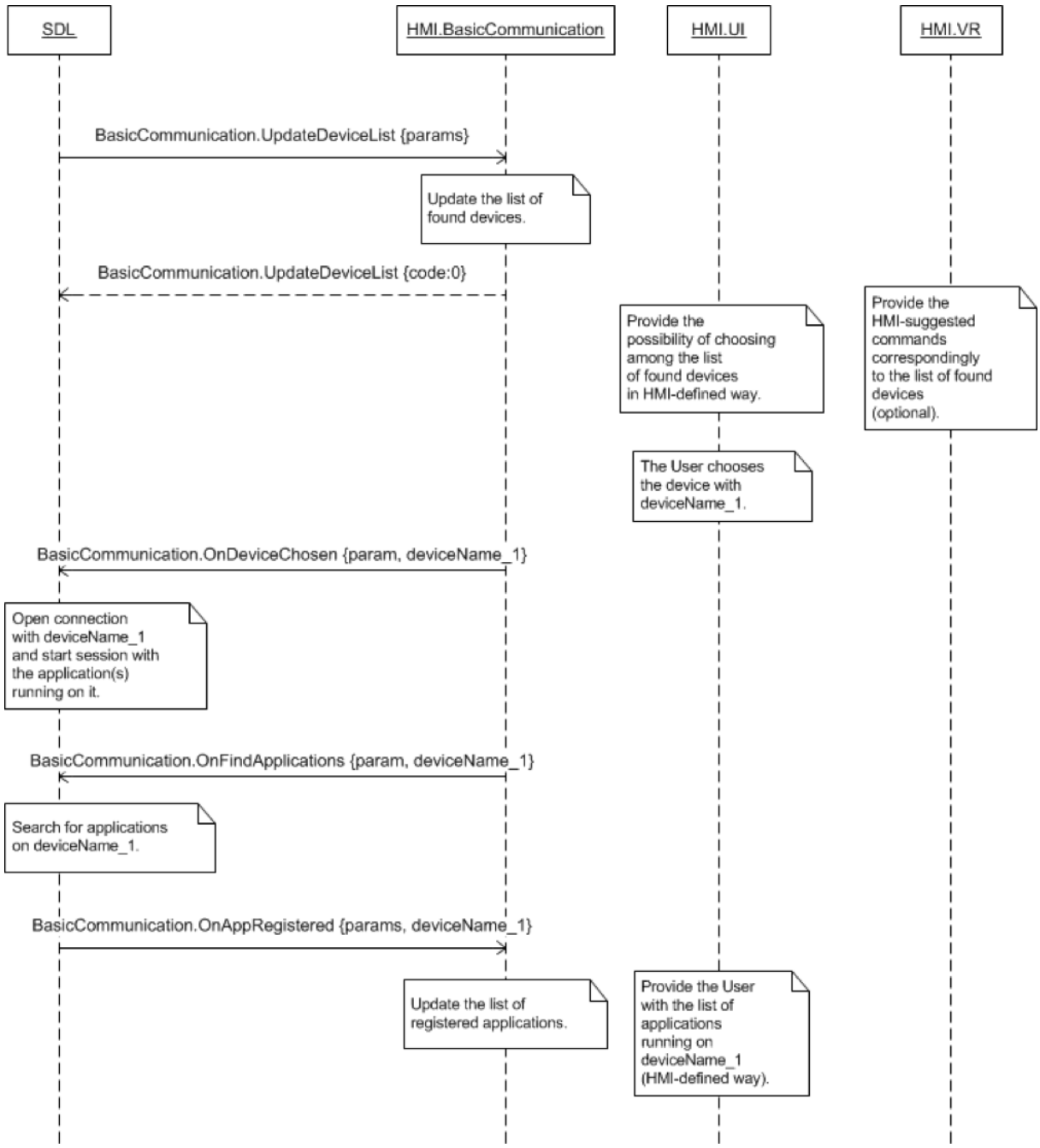
View Diagram



SEQUENCE DIAGRAM

OnFindApplications after Device Chosen by User

View Diagram



JSON EXAMPLE NOTIFICATION

```
{
 "jsonrpc" : "2.0",
 "method" : "BasicCommunication.OnFindApplications",
 "params" :
 {
  "deviceinfo" :
   {
    "name" : "XT910",
    "id" : 4
   }
 }
}
```

OnIgnitionCycleOver

Type Notification
Sender HMI
Purpose Trigger SDL to increment the counter of ignition cycles in Local Policy Table

Notification

MUST

Send `BC.OnIgnitionCycleOver` together with `BC.OnExitAllApplications(IGNITION_OFF)` to SDL once ignition off is detected.

NOTE

1. `BC.OnIgnitionCycleOver` dependencies:
2. SDL expects `BC.OnIgnitionCycleOver` *only in case* it's built with `"-DEXTENDED_POLICY: ON"` flag. *Otherwise* SDL needs `BC.OnExitAllApplications(IGNITION_OFF)` only.
3. By getting `BC.OnIgnitionCycleOver`SDL increments the `"ignition_cycles_since_last_exchange"` counter in Local Policies Table
4. Once the value of `"ignition_cycles_since_last_exchange"` gets equal to the value of `"exchange_after_x_ignition_cycles"`, SDL triggers a Policy Table Update

PARAMETERS

This RPC has no additional parameter requirements

JSON EXAMPLE NOTIFICATION

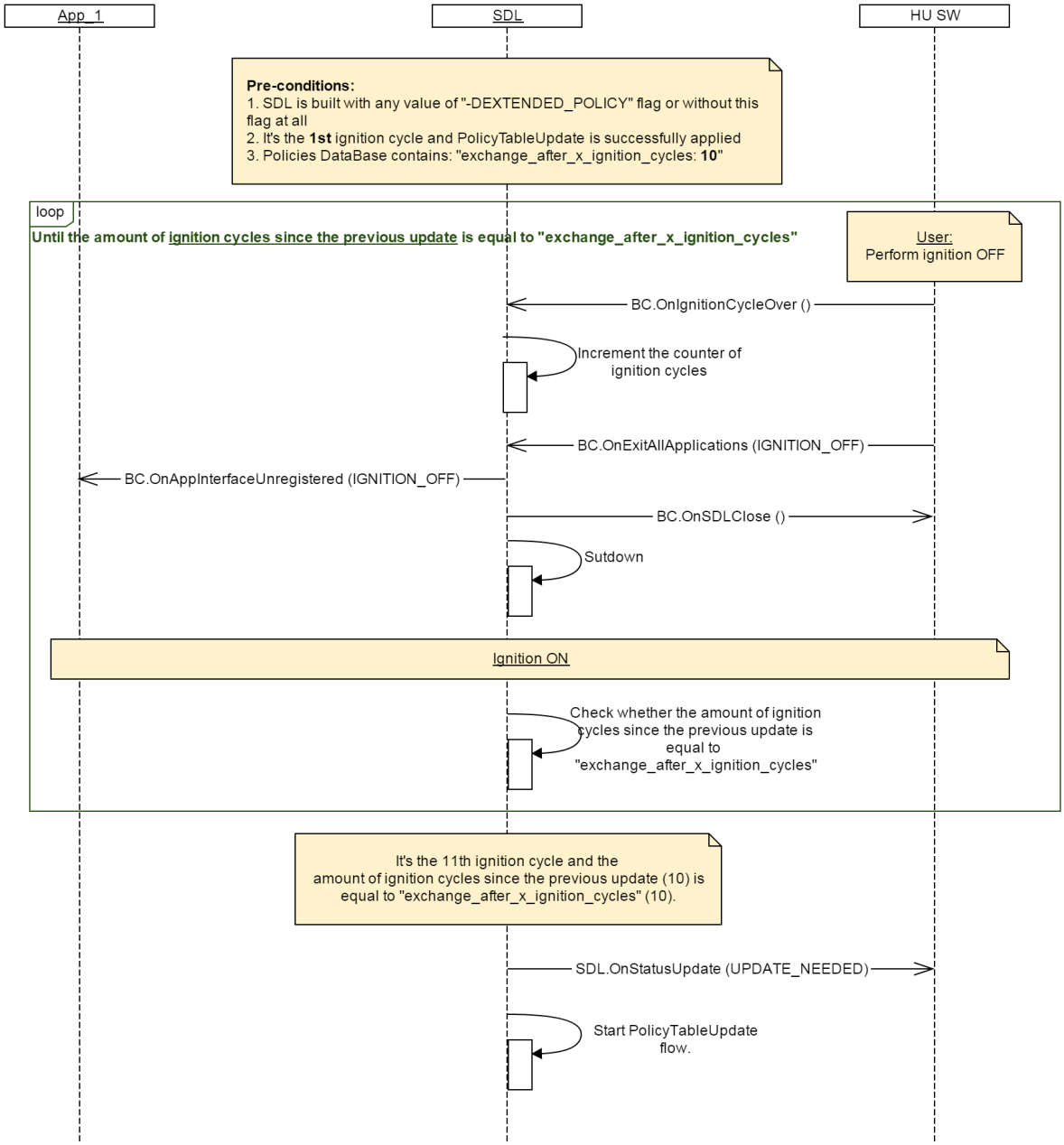
```
{
   "jsonrpc" : "2.0",
   "method" : "BasicCommunication.OnIgnitionCycleOver"
}
```

Sequence Diagrams

SEQUENCE DIAGRAM

OnIgnitionCycleOver in triggering a Policy Table Update flow

View Diagram



OnPutFile

Type

Notification

Sender

SDL

Purpose

Inform the HMI that a file has been uploaded into a shared folder by an application.

`OnPutFile` notifies the HMI that some file has been put into a shared or system folder that can be used by the HMI.

MUST

1. Use the appropriate uploaded file according to its workflow (IVSU, SystemRequest, RPC's). See diagrams listed below.
2. Whenever HMI gets RPC with `Image` which has `.isTemplate` set to true, the HMI has to:
3. load the proper image pattern
4. extract alpha channel from the template image
5. apply the alpha channel to the image pattern
6. use this newly generated image instead of the uploaded one

and whenever the UI changes, the HMI has to recreate images currently visible on the screen.

NOTE

The list of RPCs and data structures that `OnPutFile` affects are:

- Show (Image, SoftButton)
- ShowConstantTBT (Image, SoftButton)
- CreateInteractionChoiceSet (Image)
- SetGlobalProperties (Image, VrHelpItem)
- ResetGlobalProperties (Image, VrHelpItem)
- UpdateTurnList (Turn, SoftButton)
- AddCommand(Image)
- SendLocation(Image)
- Alert (SoftButton)
- AlertManeuver (SoftButton)
- ScrollableMessage (SoftButton)

Notification

PARAMETERS

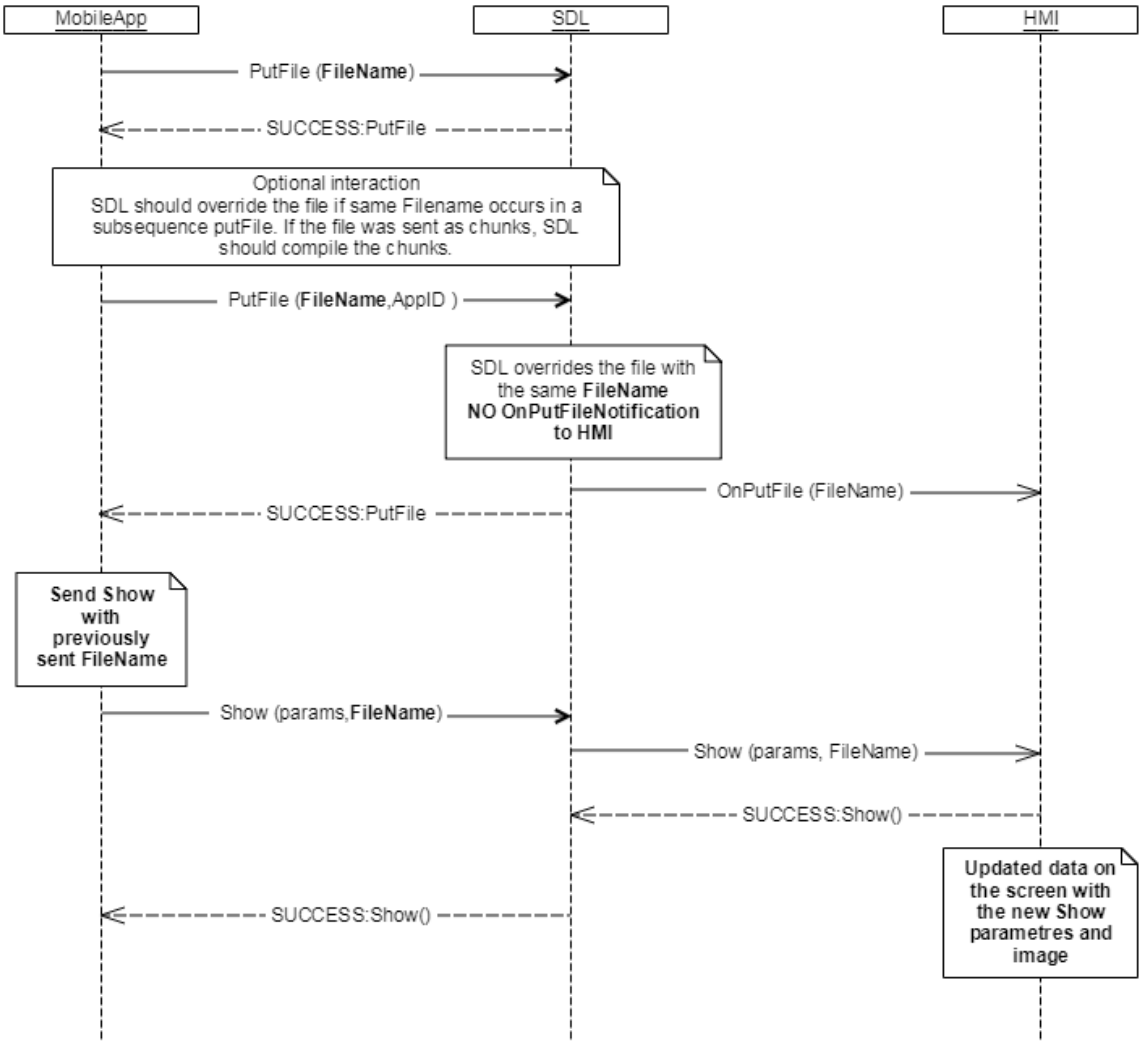
NAME	TYPE	MANDATORY	ADDITIONAL
offset	Integer	false	minvalue: 0 maxvalue: 100000000000
length	Integer	false	minvalue: 0 maxvalue: 100000000000
fileSize	Integer	false	minvalue: 0 maxvalue: 100000000000
FileName	String	true	maxlength: 255
syncFileName	String	true	maxlength: 255
fileType	Common.FileType	true	
persistentFile	Boolean	false	

Sequence Diagrams

SEQUENCE DIAGRAM

Put File used before referencing RPC

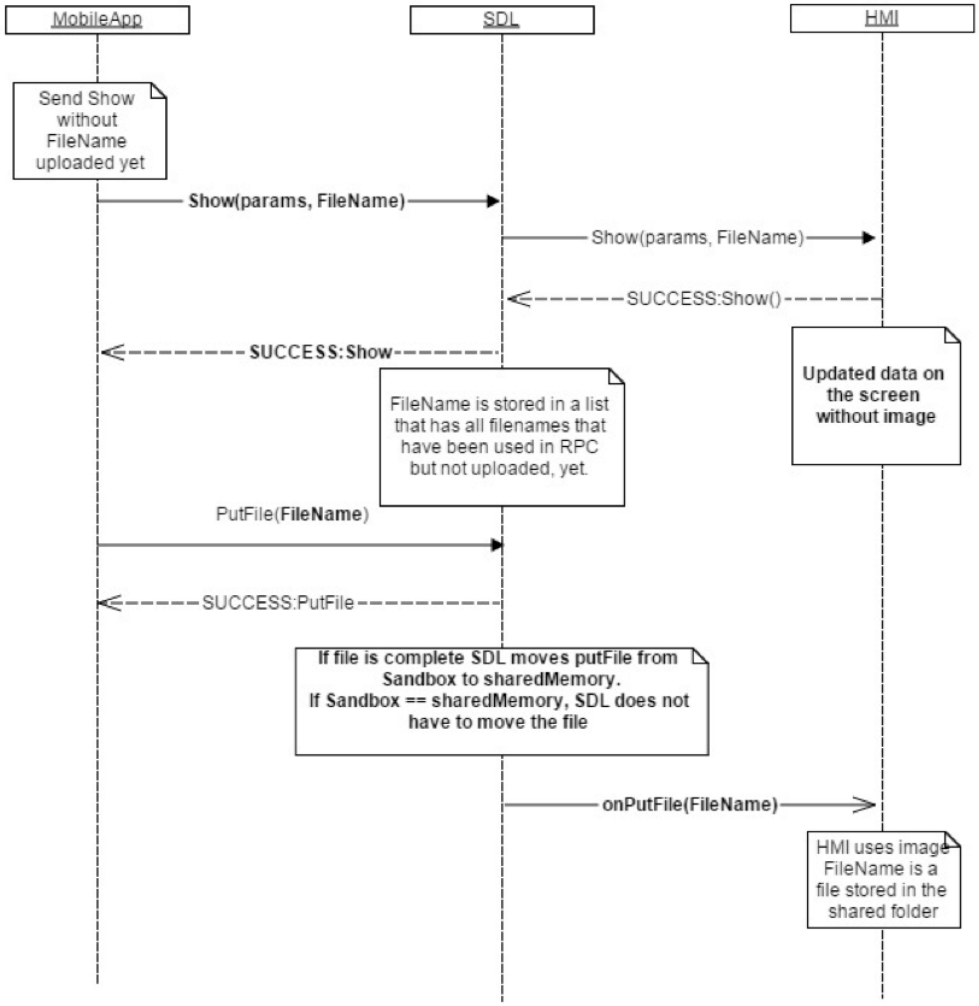
View Diagram



SEQUENCE DIAGRAM

Put File used after referencing RPC

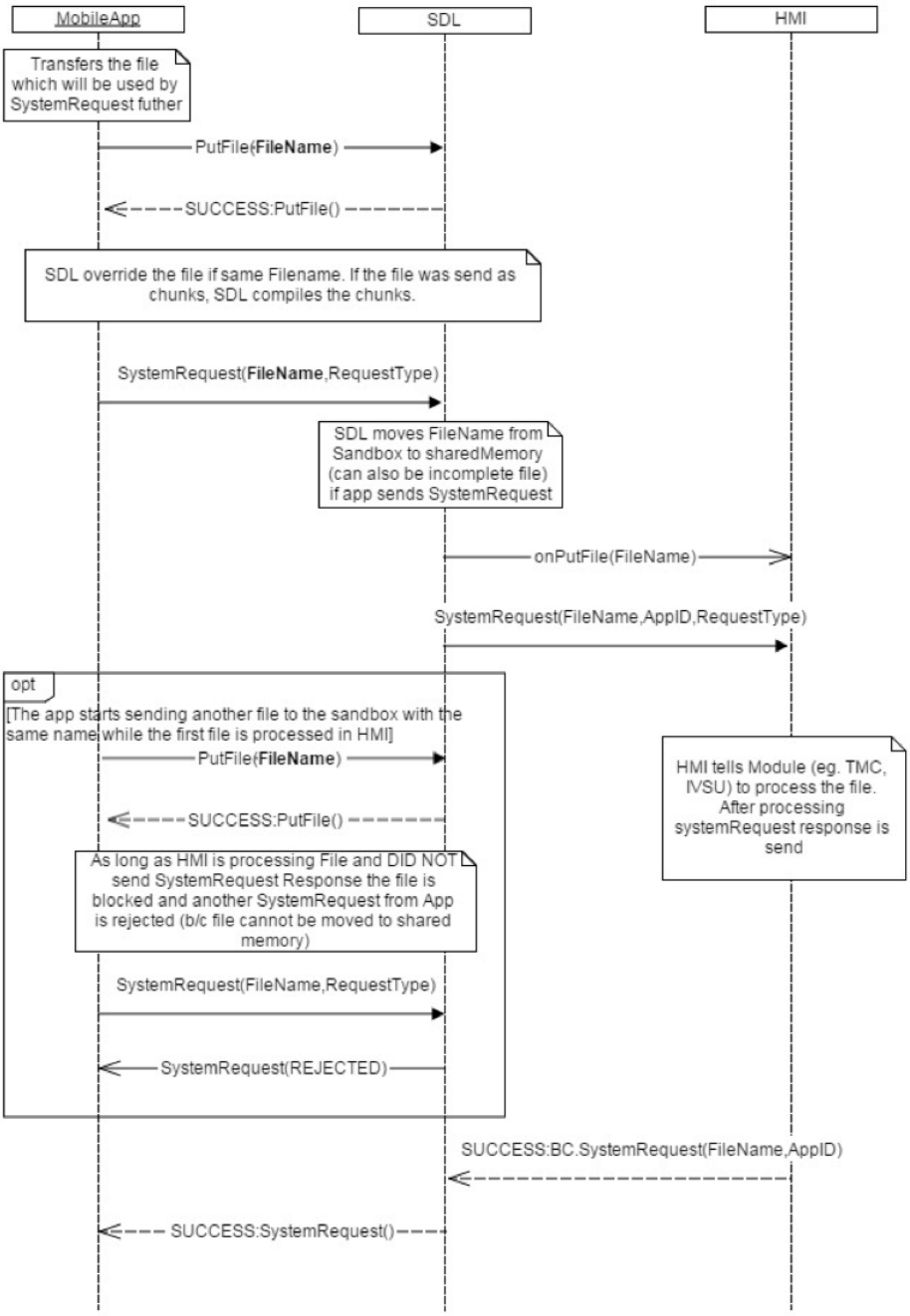
View Diagram



SEQUENCE DIAGRAM

System Request file upload using Put File

View Diagram


MobileApp
SDL
HMI
Transfers the file which will be used by SystemRequest futher
PutFile(FileName)
SUCCESS:PutFile()
SDL override the file if same Filename. If the file was send as chunks, SDL compiles the chunks.
SystemRequest(FileName,RequestType)
SDL moves FileName from Sandbox to sharedMemory (can also be incomplete file) if app sends SystemRequest
onPutFile(FileName)
SystemRequest(FileName,AppID,RequestType)
opt
[The app starts sending another file to the sandbox with the same name while the first file is processed in HMI]
PutFile(FileName)
SUCCESS:PutFile()
As long as HMI is processing File and DID NOT send SystemRequest Response the file is blocked and another SystemRequest from App is rejected (b/c file cannot be moved to shared memory)
SystemRequest(FileName,RequestType)
SystemRequest(REJECTED)
HMI tells Module (eg. TMC, IVSU) to process the file. After processing systemRequest response is send
SUCCESS:BC.SystemRequest(FileName,AppID)
SUCCESS:SystemRequest()

JSON EXAMPLE NOTIFICATION

```
{
 "jsonrpc" : "2.0",
 "method" : "BasicCommunication.OnPutFile"
 {
   "fileName":"/fs/sharedFolder/app1_device1/icon.jpg",
   "fileType":"GRAPHIC_JPEG"
  }
}
```

OnResumeAudioSource

Type Notification
Sender SDL
Purpose Inform the HMI that an application being resumed needs to become audible.

SDL sends `OnResumeAudioSource` to the HMI when the resumption process sees that a currently registered application was `AUDIBLE` before the previous `Ignition Off`.

NOTE

SDL will send `OnResumeAudioSource` if the application meets both of these conditions:

1. Application was running 30 minutes prior to OnExitAllApplications(`SUSPEND`).
2. Application reconnects no later than 30 seconds after SDL receives an OnReady notification from the HMI.

MUST

1. Activate the audio source for the application corresponding to the `appID` that was received.

2. Do not activate the application itself. The application must stay in the background of the UI.
    - If SDL means for the app to be resumed to the foreground of the UI, SDL would send an ActivateApp notification instead (See diagrams below).

Notification

PARAMETERS

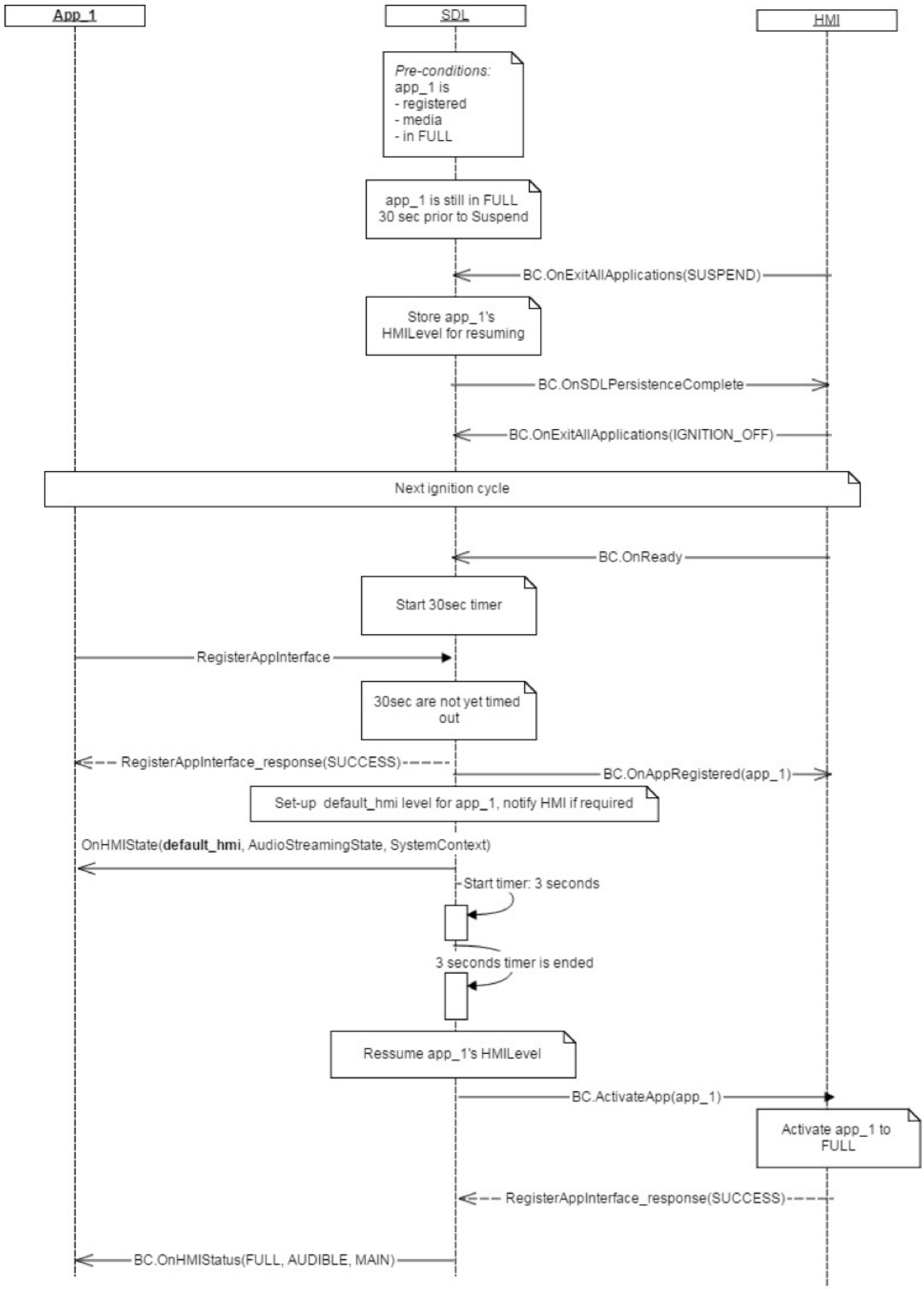
NAME	TYPE	MANDATORY	ADDITIONAL
appID	Integer	true	

Sequence Diagrams

SEQUENCE DIAGRAM

Audio Source Resumption for App in FULL

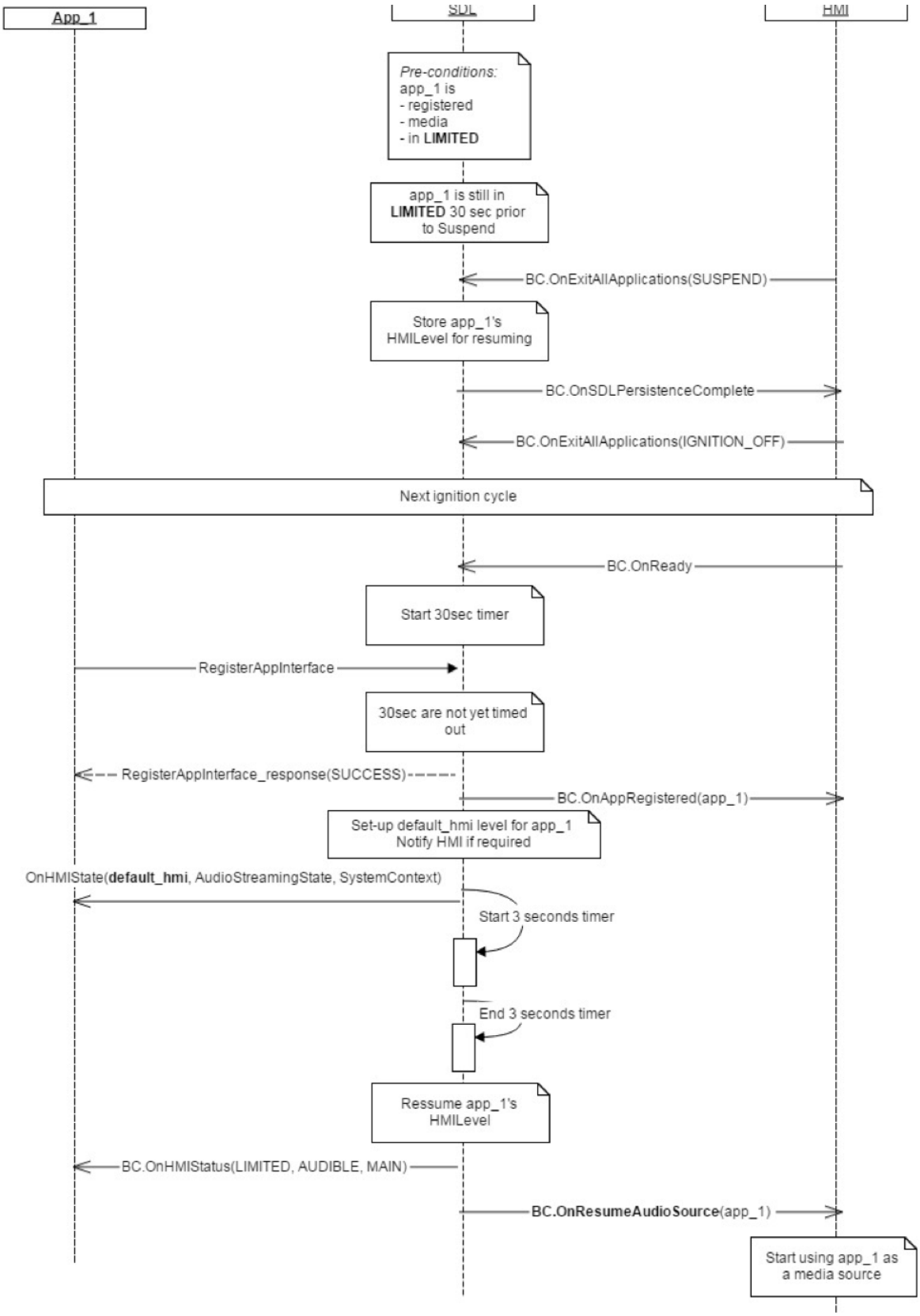
View Diagram



SEQUENCE DIAGRAM

Audio Source Resumption for App in LIMITED

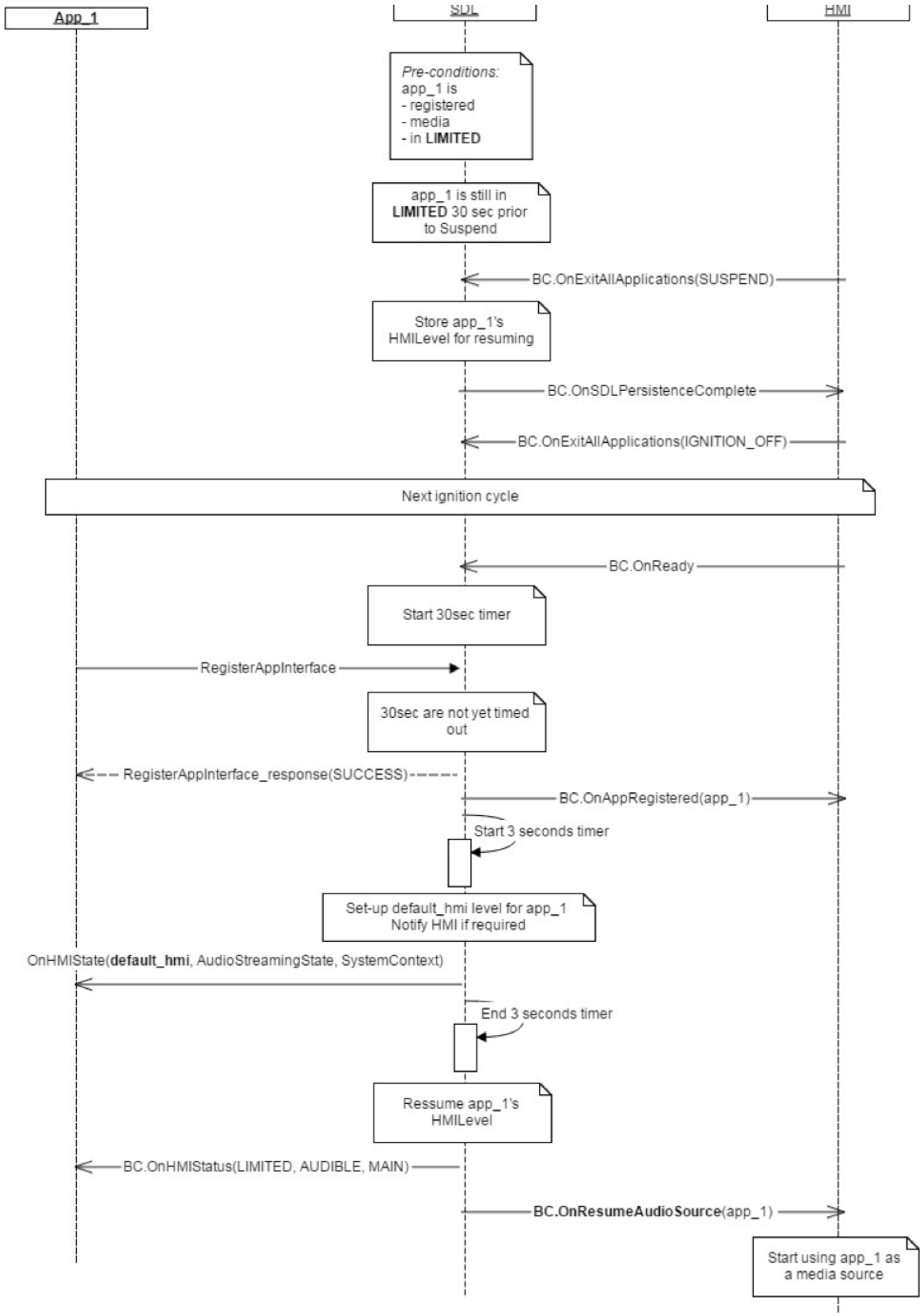
View Diagram



SEQUENCE DIAGRAM

Audio Source Resumption after Phone Call

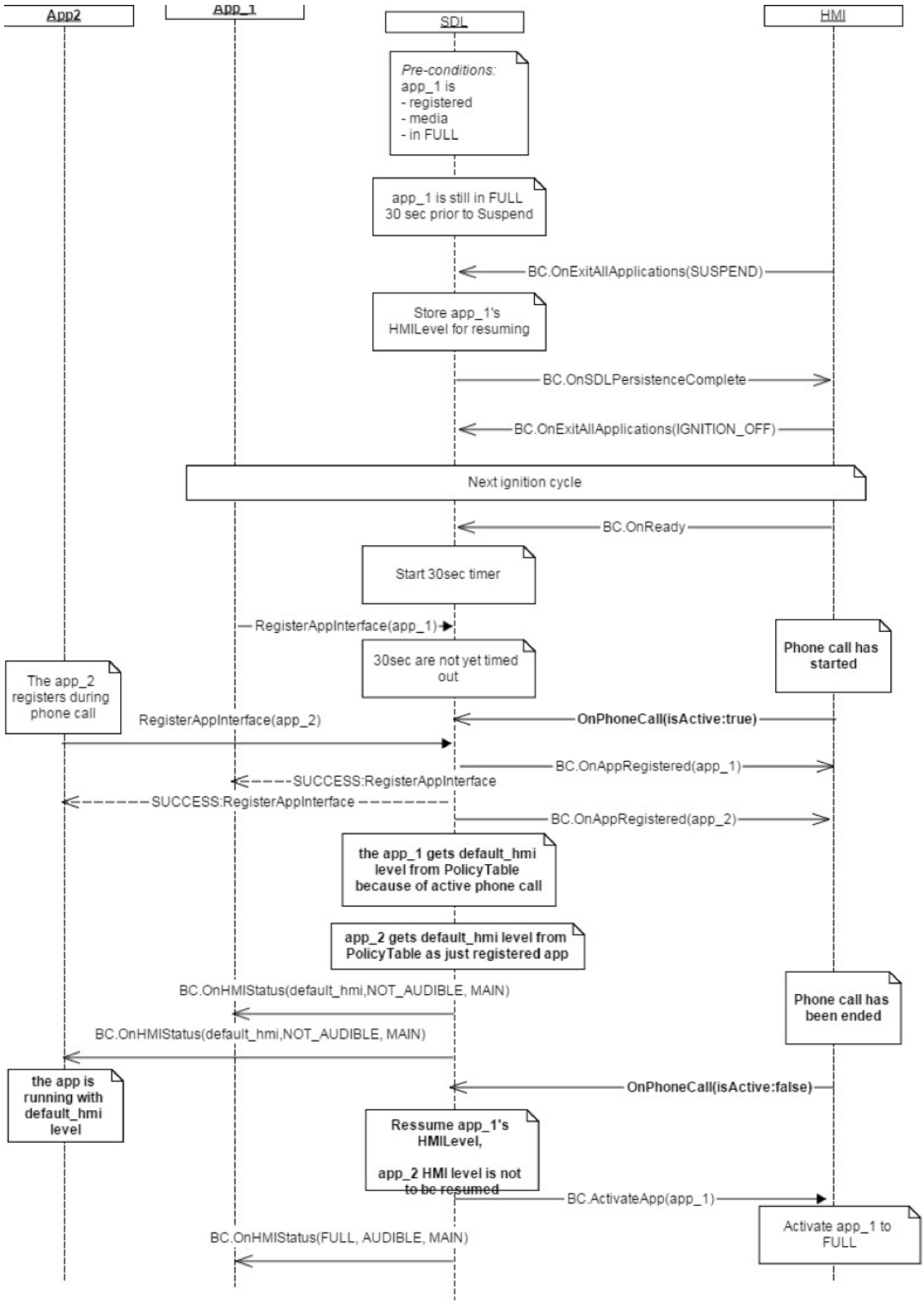
View Diagram



SEQUENCE DIAGRAM

Audio Source Resume one audio app one phone call app

View Diagram


App2
App_1
SDL
HMI
Pre-conditions:
app_1 is
- registered
- media
- in FULL
app_1 is still in FULL
30 sec prior to Suspend
BC.OnExitAllApplications(SUSPEND)
Store app_1's
HMILevel for resuming
BC.OnSDLPersistenceComplete
BC.OnExitAllApplications(IGNITION_OFF)
Next ignition cycle
BC.OnReady
Start 30sec timer
RegisterAppInterface(app_1)
30sec are not yet timed out
Phone call has started
The app_2 registers during phone call
RegisterAppInterface(app_2)
OnPhoneCall(isActive:true)
BC.OnAppRegistered(app_1)
SUCCESS:RegisterAppInterface
SUCCESS:RegisterAppInterface
BC.OnAppRegistered(app_2)
the app_1 gets default_hmi level from PolicyTable because of active phone call
app_2 gets default_hmi level from PolicyTable as just registered app
BC.OnHMIStatus(default_hmi,NOT_AUDIBLE, MAIN)
Phone call has been ended
BC.OnHMIStatus(default_hmi,NOT_AUDIBLE, MAIN)
the app is running with default_hmi level
OnPhoneCall(isActive:false)
Ressume app_1's HMILevel,
app_2 HMI level is not to be resumed
BC.ActivateApp(app_1)
Activate app_1 to FULL
BC.OnHMIStatus(FULL, AUDIBLE, MAIN)

JSON EXAMPLE NOTIFICATION

```
{
 "jsonrpc" : "2.0",
 "method" : "BasicCommunication.OnResumeAudioSource"
    "params" :
  {
    "appID" : 123
   }

}
```

OnSDLClose

Type Notification
Sender SDL
Purpose Inform the HMI that SDL is ready to be terminated by the system.

`OnSDLClose` notification is sent by SDL to notify the HMI that SDL has terminated all activity, saved/cleaned up necessary data, unregistered all applications, and the SDL Core process is ready to be closed by the system.

MUST

1. Notify SDL about an intention to terminate because of `Ignition Off`, `Master Reset`, or `Factory Reset` via OnExitAllApplications.
2. Terminate the SDL process only after receiving `OnSDLClose`. Otherwise, application data may be lost.

NOTE

SDL unregisters applications if the transports are still availble after an `Ignition Off`.

SDL cleans up applications' persistent data if the HMI calls for a `Factory Reset` or `Master Reset` via OnExitAllApplications.

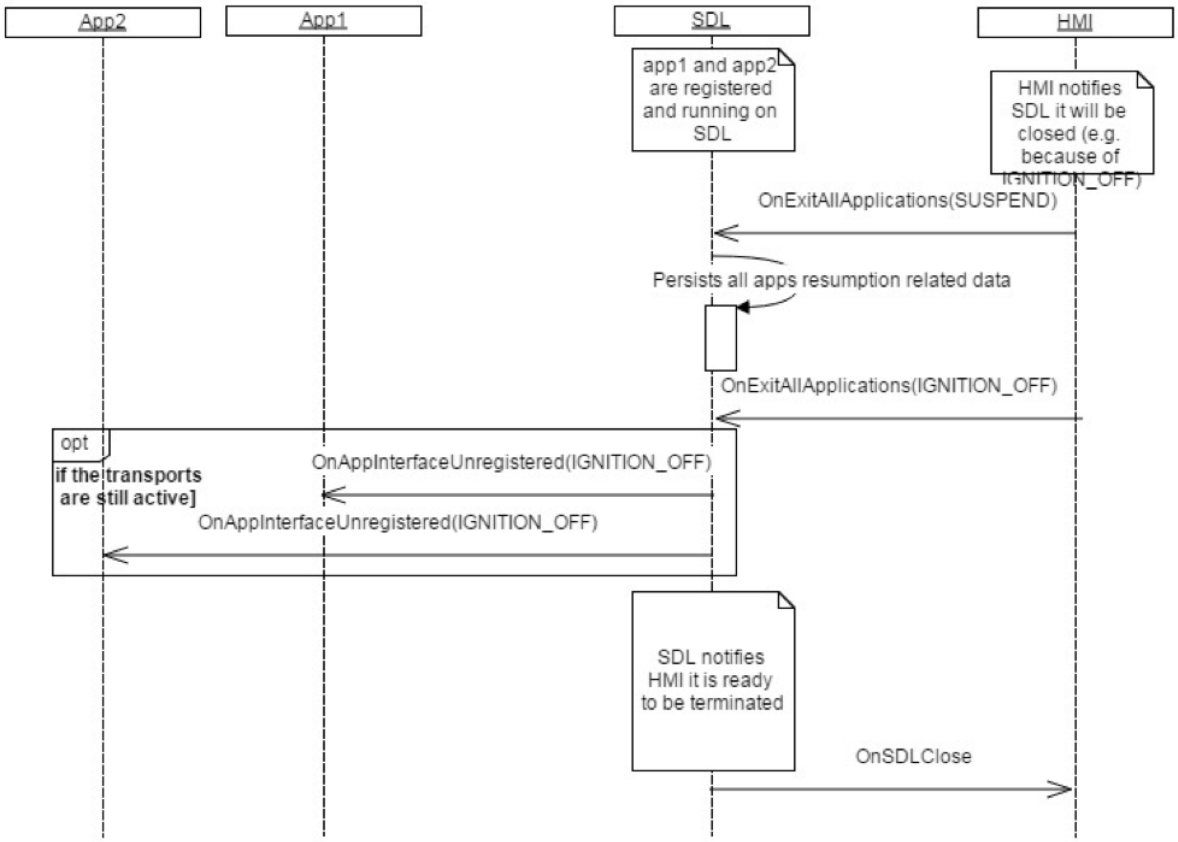
Notification

Sequence Diagrams

SEQUENCE DIAGRAM

OnSDLClose Ingition Cycle

View Diagram



SEQUENCE DIAGRAM

OnSDLClose Master Reset

View Diagram

JSON EXAMPLE NOTIFICATION

```
{
  "jsonrpc" : "2.0",
  "method" : "BasicCommunication.OnSDLClose"
}
```

OnSDLPersistenceComplete

Type
Notification
Sender
SDL
Purpose
Inform HMI that any data persistence operations have been completed.

SDL sends `OnSDLPersistenceComplete` when all data persistence operations are complete. The data made persistent is used for future resumption scenarios.

MUST

1. Send OnExitAllApplications(`SUSPEND`) to initiate the data persistence process for registered apps.

2. Wait for `OnSDLPersistenceComplete` before sending `OnExitAllApplications(IGNITION_OFF)`.

Notification

PARAMETERS

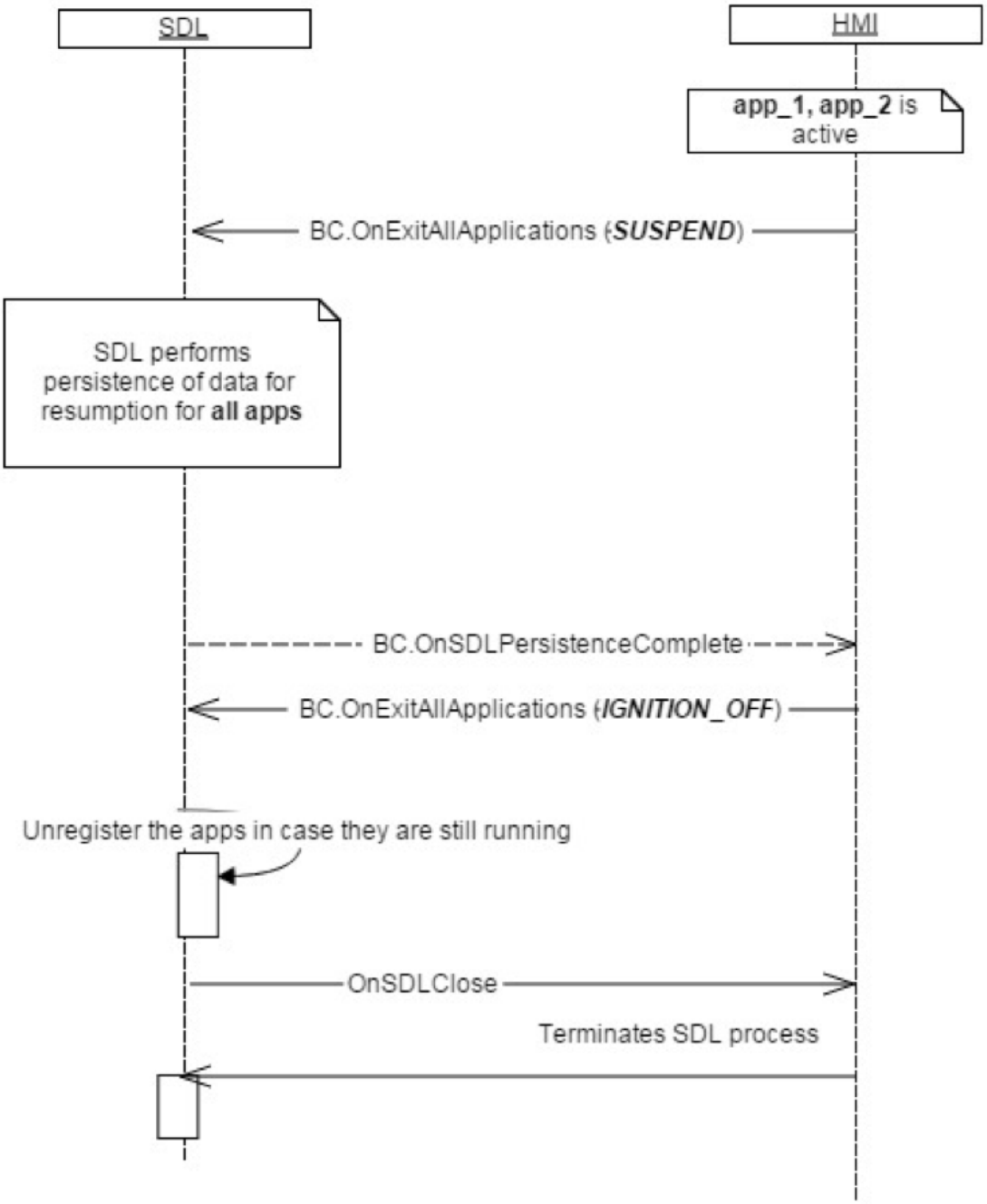
This RPC has no additional parameter requirements

Sequence Diagrams

SEQUENCE DIAGRAM

SDL Persists App Data after Suspend before Ignition Off

View Diagram


SDL
HMI
app_1, app_2 is active
BC.OnExitAllApplications (SUSPEND)
SDL performs persistence of data for resumption for all apps
BC.OnSDLPersistenceComplete
BC.OnExitAllApplications (IGNITION_OFF)
Unregister the apps in case they are still running
OnSDLClose
Terminates SDL process

JSON EXAMPLE NOTIFICATION

```
{
  "jsonrpc" : "2.0",
  "method" : "BasicCommunication.OnSDLPersistenceComplete"
}
```

OnSystemInfoChanged

Notification

PARAMETERS

PARAMETERS

NAME	TYPE	MANDATORY	ADDITIONAL
language	Common.Language	true	

JSON EXAMPLE NOTIFICATION

OnSystemRequest

Type Notification
Sender SDL
Purpose A request from SDL to download data via a connected application.

HTTP flow

- SDL sends the PTS snapshot as binary data via an `OnSystemRequest` mobile request from the system to the backend. Both the "url" that the

PTS will be forwarded to and the "timeout" must be taken from the Local Policy Table.

- If no "url" is provided in the Local Policy Table, it is supposed that the mobile application will send the Policy Table Update data back to SDL.

`RequestType` defines the type of the requested data from a mobile device or the cloud. The HMI may request this data, but it's SDL's responsibility to block a `SystemRequest` in case the request intends to transfer a data type not allowed by the policy table.
`requestSubType` is filled for supporting OEM proprietary data exchanges.

The HMI is informed about `requestTypes`, `requestSubType` that are allowed by policies via OnAppPermissionChanged and OnAppRegistered.

> NOTE
>
> If the HMI sends `OnSystemRequest` with a request type disallowed by the policy table, SDL will ignore it.
> In case PT contains some value for `requestSubType` param and the HMI sends `OnSystemRequest` with `requestSubType=<not_in_PT>`, SDL does not forward this notification to mobile app.

SyncP NOTE
1. It's SyncP responsibility to encrypt and encode PTS file and provide it to SDL via OnSystemRequest HMI API ("filename") parameter.
2. It's SyncP responsibility to choose an application for sending PTU and start timer (for future retry strategy) after sending OnSystemRequest to SDL.

MUST

HMI must send `OnSystemRequest`, if specific data is requested from the mobile device/cloud, or binary data needs to be sent to the mobile device.

Notification

PARAMETERS

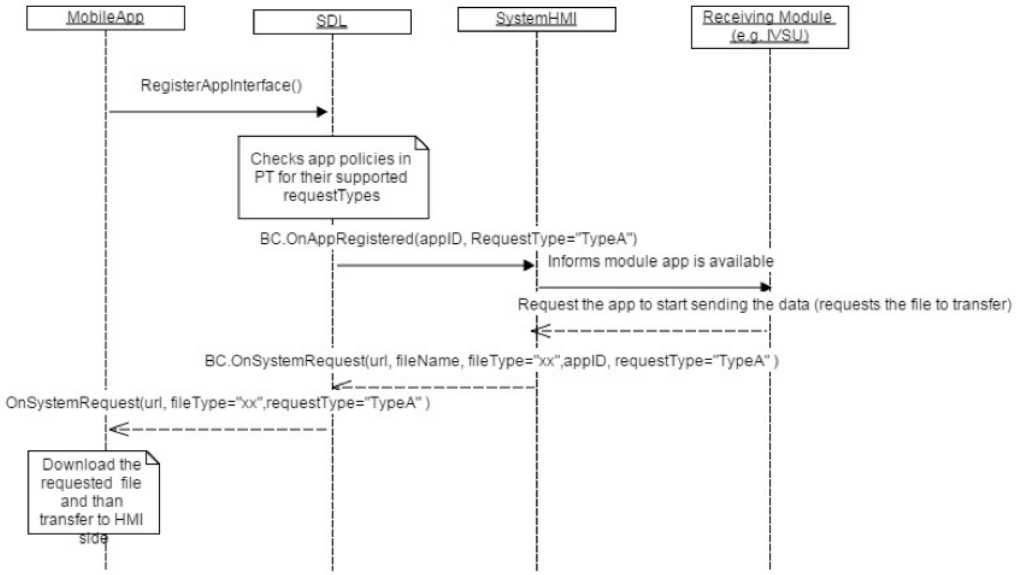
NAME	TYPE	MANDATORY	ADDITIONAL
requestType	Common.RequestType	true	
requestSubType	String	false	maxlength: 255
url	String	false	minlength: 1 maxlength: 1000
fileType	Common.FileType	false	
offset	Integer	false	minvalue: 0 maxvalue: 100000000000
length	Integer	false	minvalue: 0 maxvalue: 100000000000
timeout	Integer	false	minvalue: 0 maxvalue: 2000000000
fileName	String	true	minlength: 1 maxlength: 255
appID	String	false	minlength: 1 maxlength: 50

Sequence Diagrams

SEQUENCE DIAGRAM

System Requests File Download

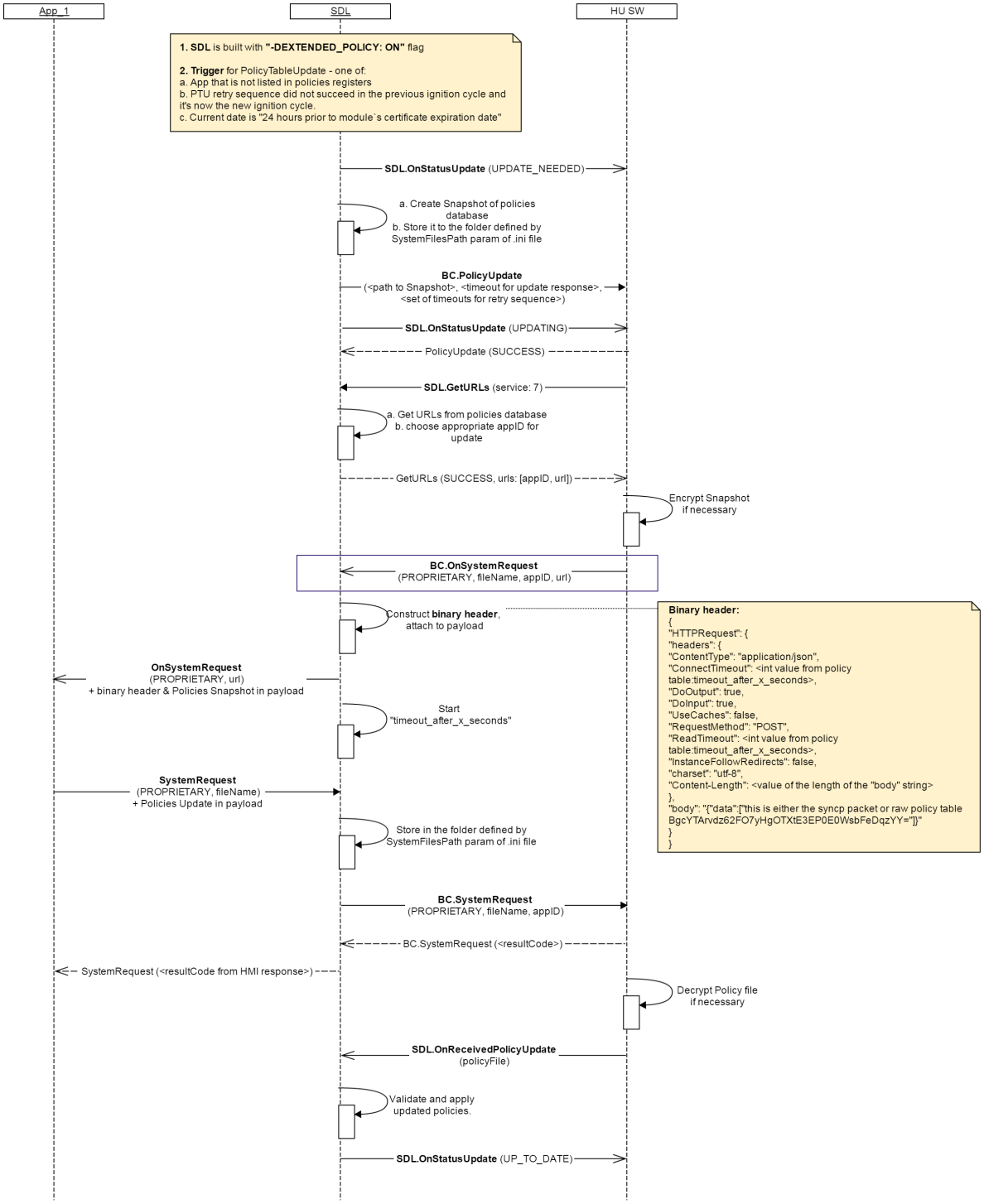
View Diagram



SEQUENCE DIAGRAM

BC.OnSystemRequest in "Proprietary" Policy Table Update Flow

View Diagram



SEQUENCE DIAGRAM

BC.OnSystemRequest in External Proprietary Policy Table Update Flow

View Diagram

JSON EXAMPLE NOTIFICATION

```
{
 "jsonrpc" : "2.0",
 "method" : "BasicCommunication.OnSystemRequest",
 "params" :
 {
  "fileName":"/fs/images/ivsu_cache/EncodedPolicyTable.json",
   "fileType":"JSON",
   "length":0,
   "offset":0,
   "requestType":"PROPRIETARY",
   "timeout":1000,
   "url":"https://policies.smartdevicelink.org/api/1/policies"
  }
}
```

OnUpdateDeviceList

Type Notification
Sender HMI

Purpose

Ask for the updated list of connected devices.

Sending an `OnUpdateDeviceList` notification to SDL asks SDL to send back a list of discovered devices via UpdateDeviceList.

MAY

1. Provide the user with the possibility to request for an updated list of discovered devices.
2. Send `OnUpdateDeviceList` when the user has chosen to update the device list.

!!! NOTE

`OnUpdateDeviceList` is similar and can be confused with `OnStartDeviceDiscovery`. The difference is that `OnUpdateDeviceList` asks SDL to return an updated list of discovered devices, and `OnStartDeviceDiscovery` asks SDL to initiate a search procedure to discover new devices.

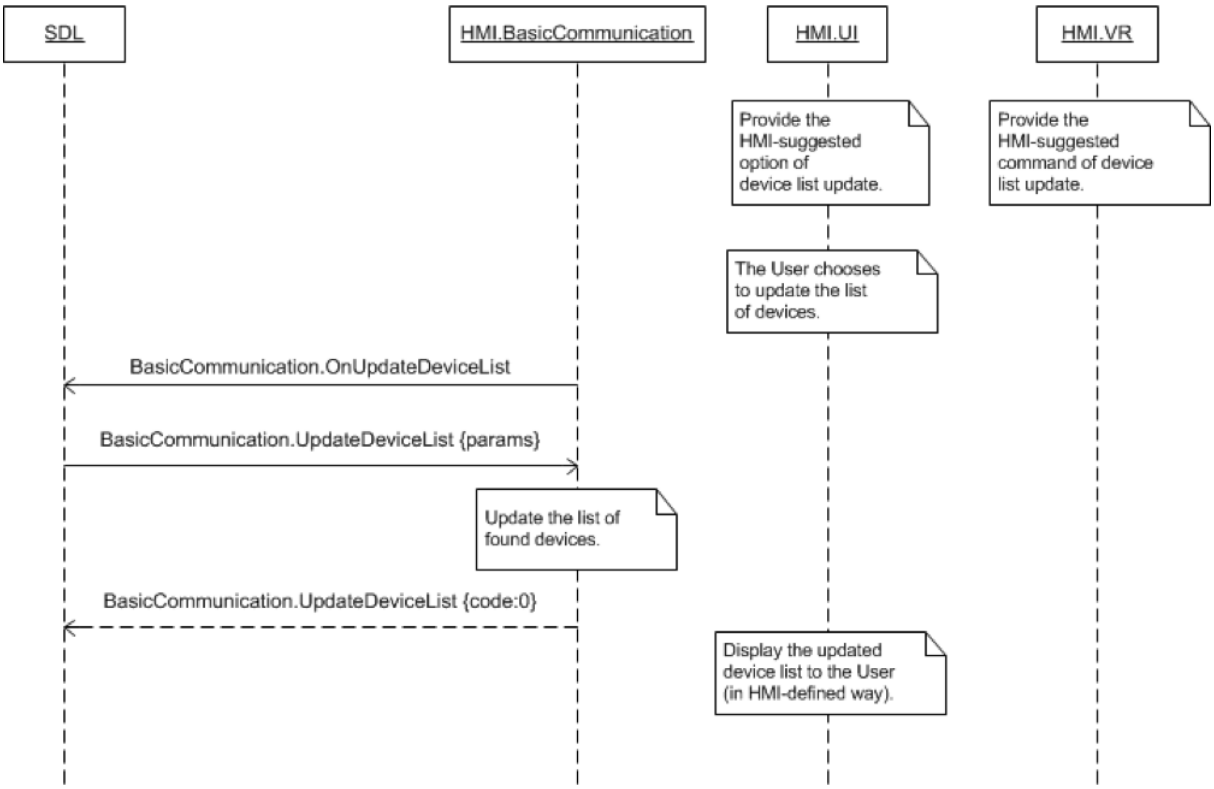
Notification

Sequence Diagrams

SEQUENCE DIAGRAM

User Requests Device List Update

View Diagram



JSON EXAMPLE NOTIFICATION

```
{
  "jsonrpc" : "2.0",
  "method" : "BasicCommunication.OnUpdateDeviceList"
}
```

PolicyUpdate

Type

Function
Sender SDL
Purpose Inform HMI about the Policy Table Update (PTU) mechanism is triggered on SDL

In case SDL is built with **"-DEXTENDED_POLICY: HTTP" flag**, SDL supports PolicyTableUpdate flow **without HMI-related logic**.

REQUEST

`BC.PolicyUpdate` represents SDL-generated request to start the PTU sequence.

MUST

Proprietary Policies:

- Encrypt the Snapshot PT (path from `file` parameter) *in case* and by the scheme required by Policies Server
- Request Policies Server url via the next `SDL.GetURLs` from SDL
- Provide the path defined by `file` parameter in the next `BC.OnSystemRequest` that SDL will forward to mobile application
- Recognize the PTU status notifications of `SDL.OnStatusUpdate` from SDL and display them in the appropriate UI menu

NOTE

1. `BC.PolicyUpdate` dependencies:
2. SDL sends `BC.PolicyUpdate` *only in case* it's built with "-DEXTENDED_POLICY: PROPRIETARY" flag or without this flag. *Otherwise* SDL handles the entire PTU flow by itself.
3. If HMI fails to respond `BC.PolicyUpdate` or responds with error, PTU sequence will *not* be continued.
4. Triggers for sending `BC.PolicyUpdate` (whichever comes first):
5. Days since previous successful PTU (`"exchange_after_x_days"` value in local PolicyTable (PT)
6. Kilometers since previous successful PTU (`"exchange_after_x_kilometers"` value in local PT)
7. Ignition cycles since previous successful PTU (`"exchange_after_x_ignition_cycles"` value in local PT)
8. 24 hours prior to module's certificate expiration date:
   a. The triggers for checking the cert expiration status are:
   Ignition On
   TLS handshake
9. New application (that is, not-yet existing in local PT) registration
10. In case the status of PTU is UPDATE_NEEDED due to failed retry stratery at previous ignition cycle
11. Parameters values origin:
12. `file` - is the path to the Snapshot of local PolicyTable (Snapshot PT final destination is Policies Server)
13. `timeout` - value taken from `"timeout_after_x_seconds"` field of local PT
14. `retry` - array of values from `"seconds_between_retries"` field of local PT. SDL handles the PTU retry sequence (re-requesting update if fails to receive during timeout) by itself.
15. When SDL is built with EXTERNAL_PROPRIETARY flow, SDL *PoliciesManager* must change the status to “UPDATING” and notify HMI with OnStatusUpdate("UPDATING") right after SnapshotPT is sent out to to mobile app via OnSystemRequest() RPC.

PARAMETERS

NAME	TYPE	MANDATORY	ADDITIONAL	DESCRIPTION
file	String	true	minlength: 1 maxlength: 255	Location of policy table snapshot. It’s defined in smartDeviceLink.ini as “PathToSnapshot” parameter
timeout	Integer	true	minvalue: 0 maxvalue: 65535	Send attempt timeout in seconds, it’s a value from the Policy Table.
retry	Integer	true	array: true minsize: 1 maxsize: 5 minvalue: 0 maxvalue: 65535	Array of delays to wait after failed atempts, it’s a value from the Policy Table

RESPONSE

MUST

Respond with `SUCCESS` resultCode to continue the PTU flow.

PARAMETERS

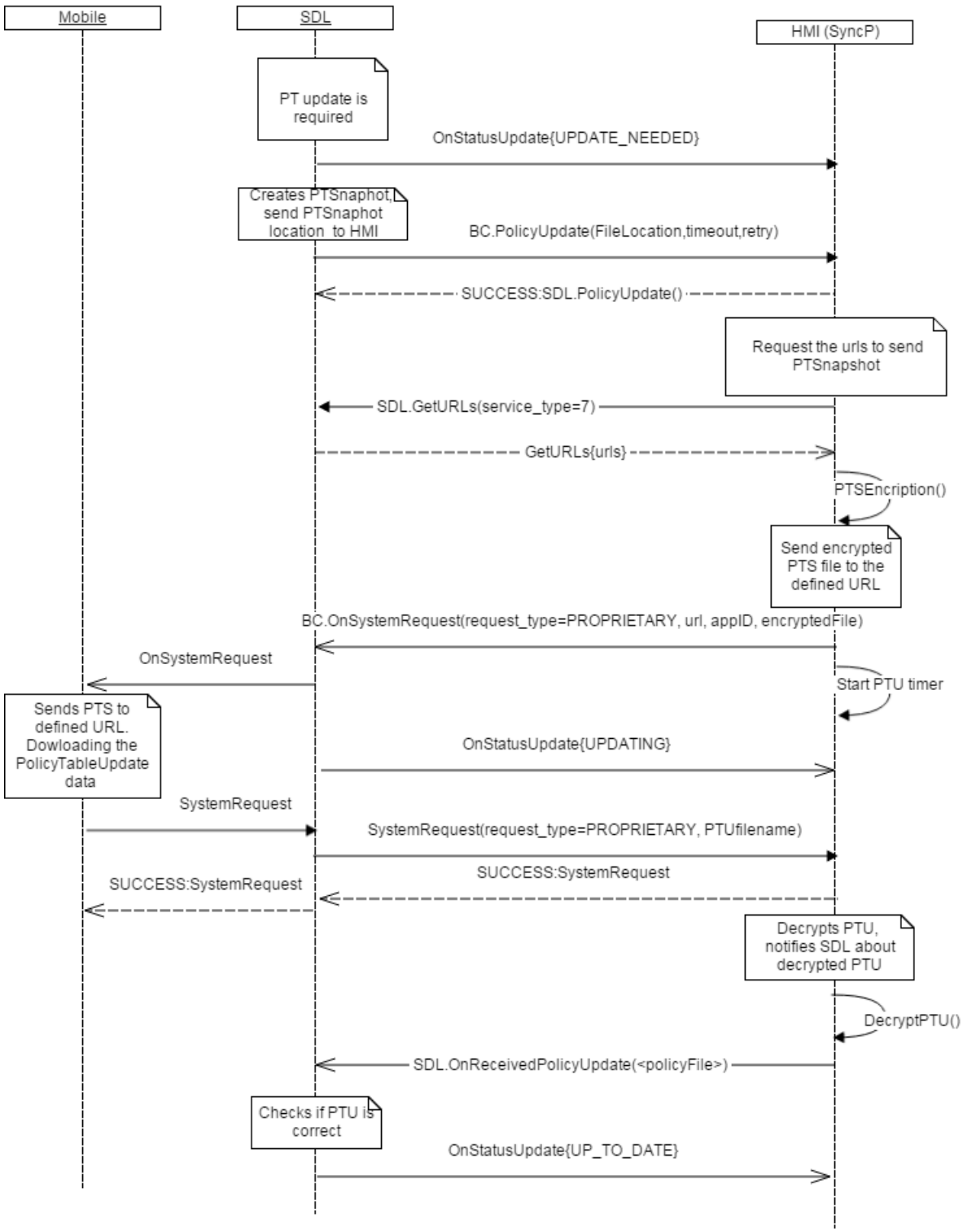
This RPC has no additional parameter requirements

Sequence Diagrams

SEQUENCE DIAGRAM

BC.PolicyUpdate in EXTERNAL PROPRIETARY Policy Table Update Flow

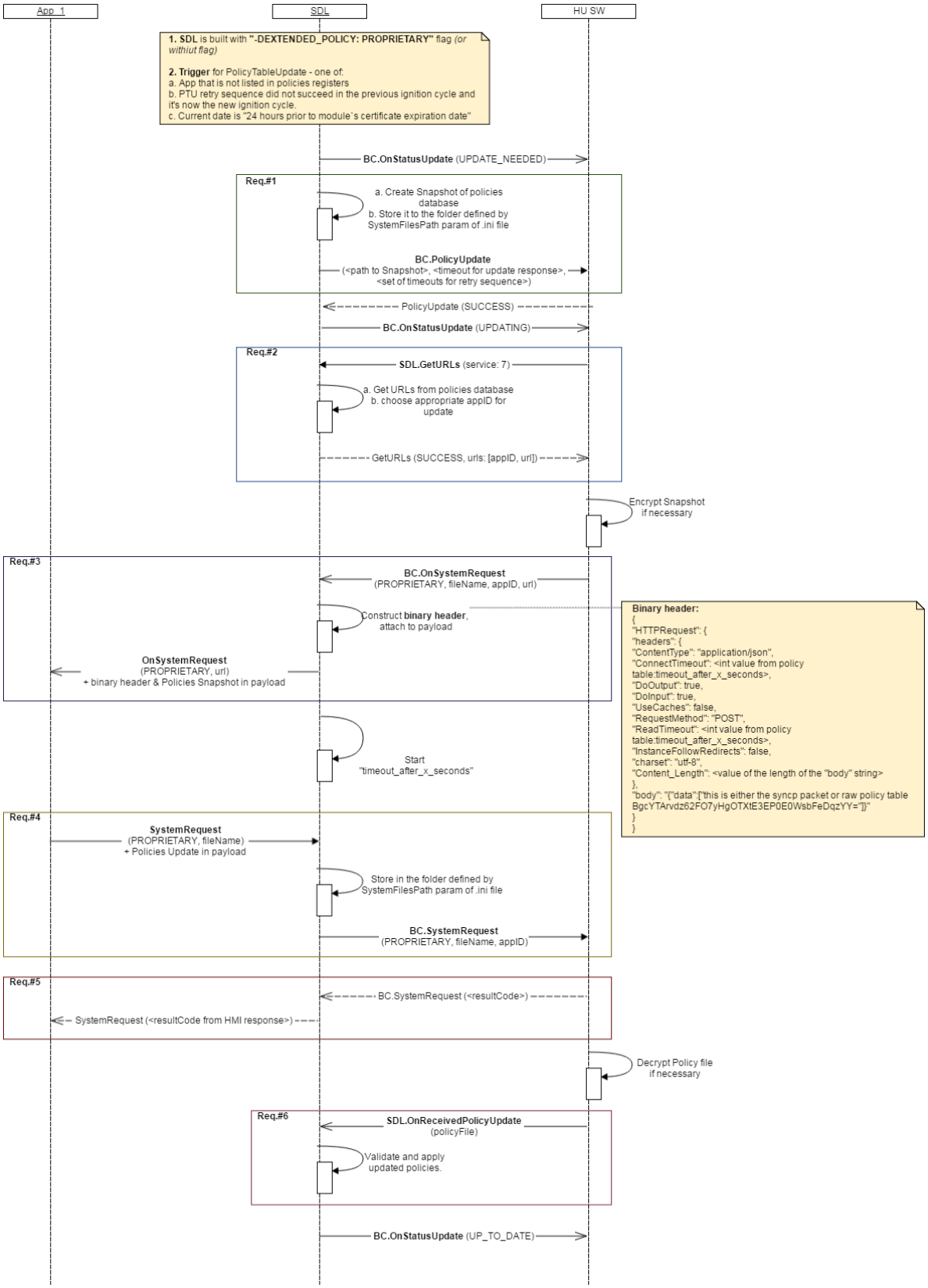
View Diagram



SEQUENCE DIAGRAM

BC.PolicyUpdate in PROPRIETARY Policy Table Update Flow

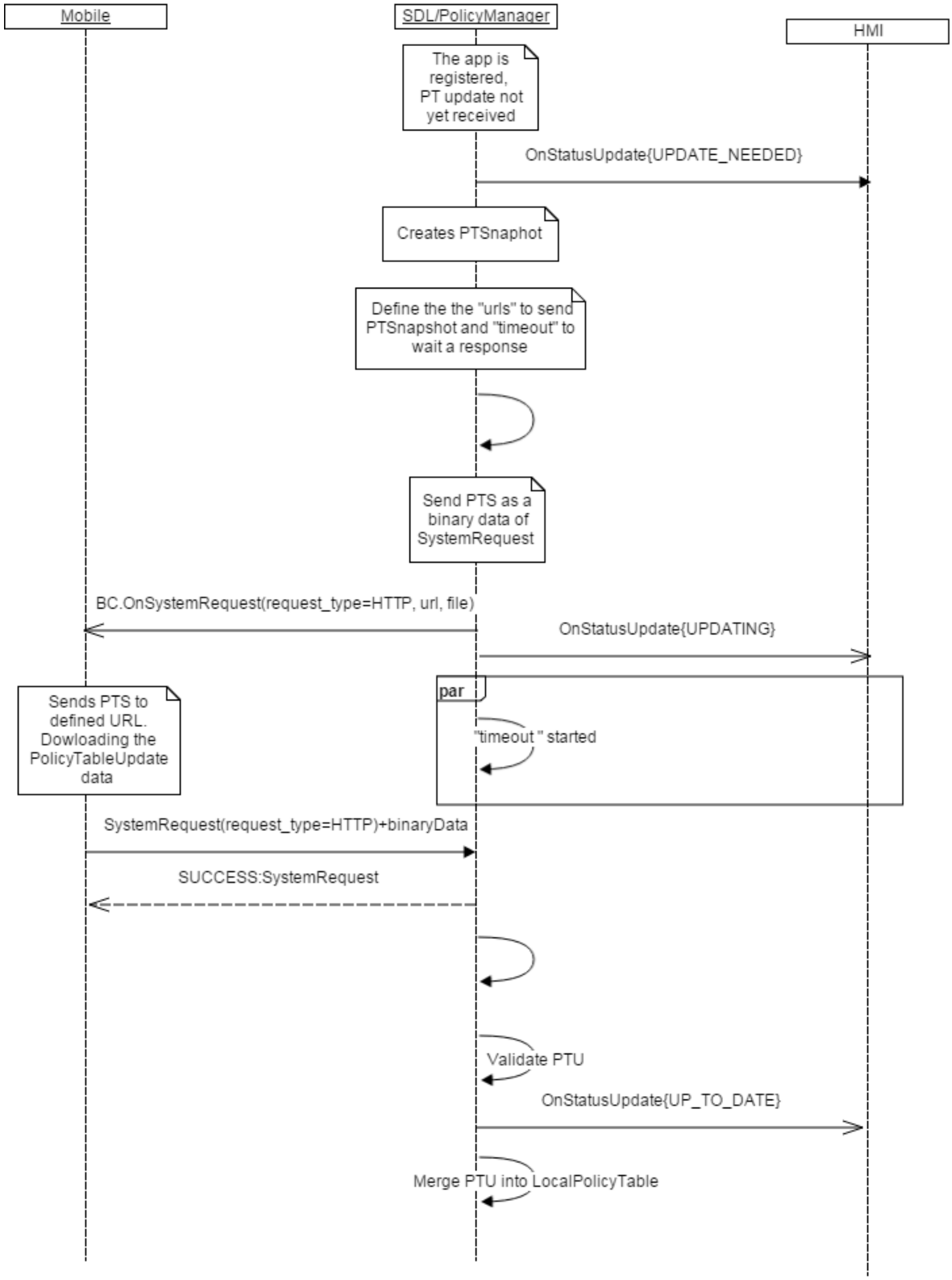
View Diagram



SEQUENCE DIAGRAM

BC.PolicyUpdate in "HTTP" Policy Table Update Flow

View Diagram


Mobile
SDL/PolicyManager
HMI
The app is registered, PT update not yet received
OnStatusUpdate{UPDATE_NEEDED}
Creates PTSnaphot
Define the the "urls" to send PTSnapshot and "timeout" to wait a response
Send PTS as a binary data of SystemRequest
BC.OnSystemRequest(request_type=HTTP, url, file)
OnStatusUpdate{UPDATING}
par
"timeout " started
Sends PTS to defined URL. Dowloading the PolicyTableUpdate data
SystemRequest(request_type=HTTP)+binaryData
SUCCESS:SystemRequest
Validate PTU
OnStatusUpdate{UP_TO_DATE}
Merge PTU into LocalPolicyTable

Example Request

```
{
   "id": 103,
   "jsonrpc": "2.0",
   "method": "BasicCommunication.PolicyUpdate",
   "params":
   {
      "file": " / tmp / fs / mp / SnapshotPT.json ",
      "timeout": 60,
      "retry": [1, 5, 25, 125, 625]
   }
}
```

Example Response

```
{
   "id": 103,
   "jsonrpc": "2.0",
   "result":
   {
      "code": 0,
      "method": "BasicCommunication.PolicyUpdate"
   }
}
```

Example Error


```
{
   "id": 103,
   "jsonrpc": "2.0",
   "error": {
      "code": 11,
      "message": "Snapshot PT file not found",
      "data":
      {
         "method": "BasicCommunication.PolicyUpdate"
      }
   }
}
```


UpdateAppList

Type Function
Sender SDL
Purpose Update HMI's list of registered applications.

SDL may send this request after connecting to a device, and an application successfully registers itself with SDL. This request may also be sent by SDL if the user requests for an update via OnFindApplications.

MUST

1. Update the list of registered applications.
2. Use the `appID` provided with this request when sending definite requests or notifications back to a mobile application (e.g. ActivateApp, OnCommand).
3. Provide the user with the possibility to choose among a list of registered applications.

NOTE

SDL adds the VR synonyms of a registered application to the HMI via OnAppRegistered.

NOTE

Cloud apps will appear in the app list before they are connected and registered. The `cloudConnectionStatus` parameter in Common.HMIApplication can be used to track the connection status of the cloud app. The `isCloudApp` parameter can be used by the HMI to differentiate or group cloud apps away from mobile connected SDL applications.

The connection to the cloud app will be created by Core after the user activates the app via SDL.ActivateApp.

REQUEST

PARAMETERS

NAME	TYPE	MANDATORY	ADDITIONAL
applications	Common.HMIApplication	true	array: true minsize: 0 maxsize: 100

RESPONSE

PARAMETERS

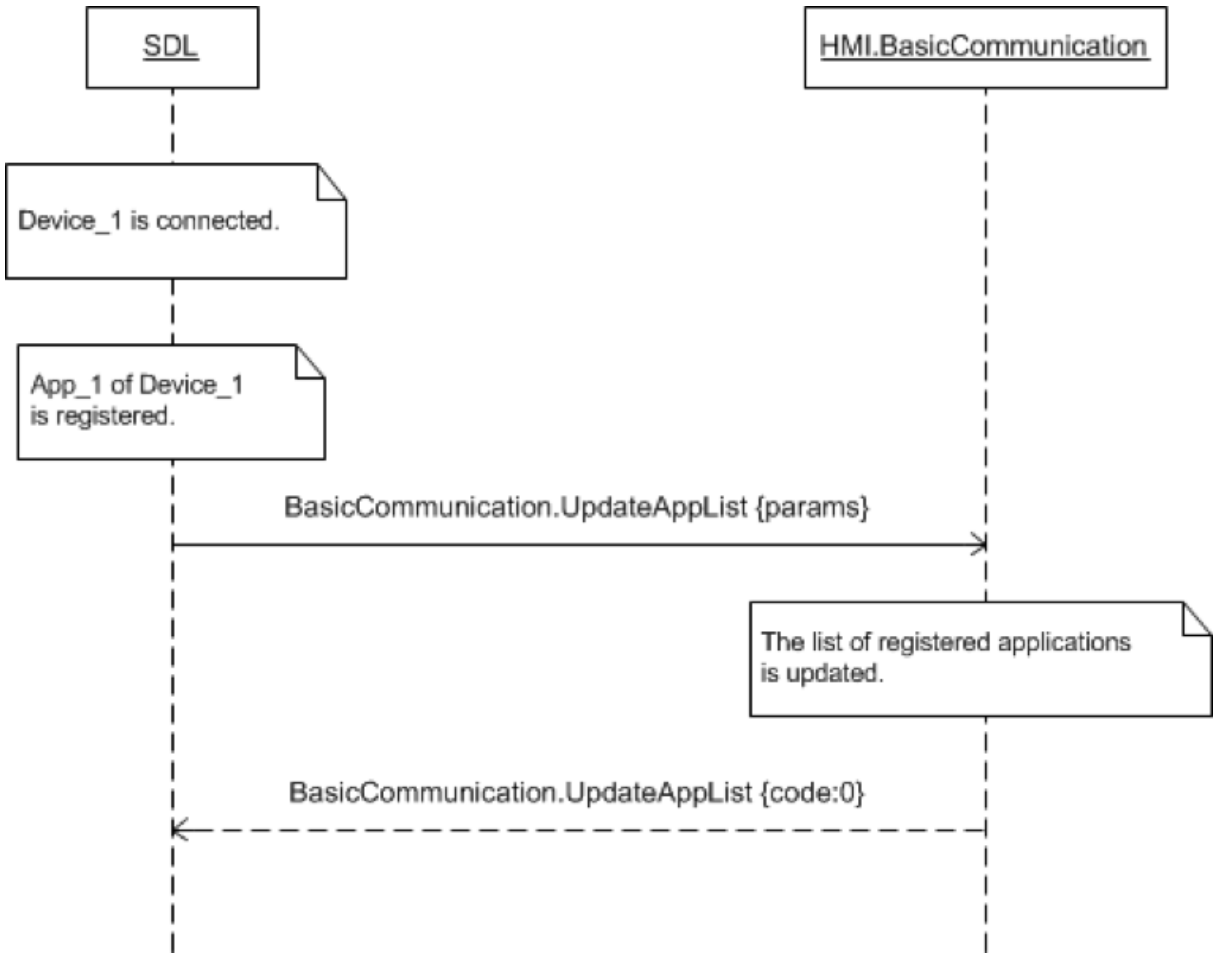
This RPC has no additional parameter requirements

Sequence Diagrams

SEQUENCE DIAGRAM

Application Just Registered

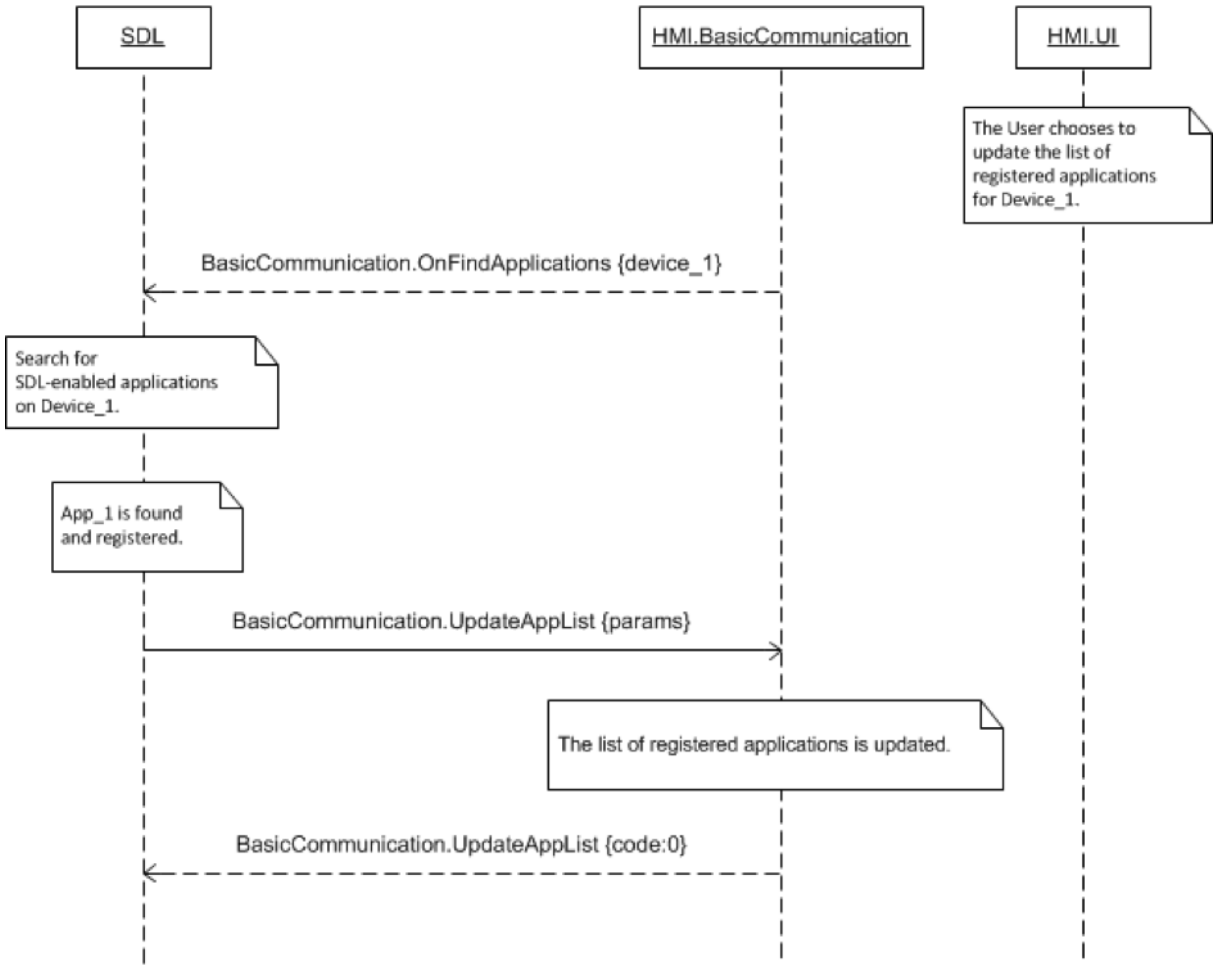
View Diagram



SEQUENCE DIAGRAM

User Requests Update App List

View Diagram


SDL
HMI.BasicCommunication
HMI.UI
The User chooses to update the list of registered applications for Device_1.
BasicCommunication.OnFindApplications {device_1}
Search for SDL-enabled applications on Device_1.
App_1 is found and registered.
BasicCommunication.UpdateAppList {params}
The list of registered applications is updated.
BasicCommunication.UpdateAppList {code:0}

Example Request

```
{
 "id" : 75,
 "jsonrpc" : "2.0",
 "method" : "BasicCommunication.UpdateAppList",
 "params" :
 {
  "applications" :
  [
   {
    "appName" : "Beautiful Sound",
    "ngnMediaScreenAppName " : "BeauSo",
    "deviceName" : "Jerry`s Phone",
    "appID" : 65544,
    "hmiDisplayLanguageDesired" : "DE-DE",
    "isMediaApplication" : true
   },
   {
    "appName" : "Go Travel",
    "icon" : "tmp/SDL/app/Go_Travel/icon.png",
    "deviceName" : "XT910",
    "appID" : 65545,
    "hmiDisplayLanguageDesired" : "EN-US",
    "isMediaApplication" : false,
    "appType" : "INFORMATION"
   },
   {
    "appName" : "Cloud App",
    "icon" : "tmp/SDL/app/cloud/icon.png",
    "appID" : 65543,
    "cloudConnectionStatus": "NOT_CONNECTED",
    "isCloudApplication": true,
    "deviceInfo": {
     "id": "a3cadf8a",
     "isSDLAllowed": true,
     "name": "ws://192.168.1.1:3000/",
     "transportType": "CLOUD_WEBSOCKET"
    }
   }
  ]
 }
}
```

Example Response


```
{
 "id" : 75,
 "jsonrpc" : "2.0",
 "result" :
 {
  "code" : 0,
  "method" : "BasicCommunication.UpdateAppList"
 }
}
```


Example Error


```
{
 "id" : 75,
 "jsonrpc" : "2.0",
 "error" :
 {
  "code" : 22,
  "message" : "During the API call an unknown error has occurred.",
  "data" :
  {
   "method" : "BasicCommunication.UpdateAppList"
  }
 }
}
```

OnSystemTimeReady

Type Notification
Sender HMI
Purpose Notifies SDL that the HMI is ready to provide accurate system time

MUST

The HMI must send a BC.OnSystemTimeReady notification to SDL when a new ignition cycle occurs in order to notify SDL that the HMI is once again ready to provide accurate system time.

Notification

PARAMETERS

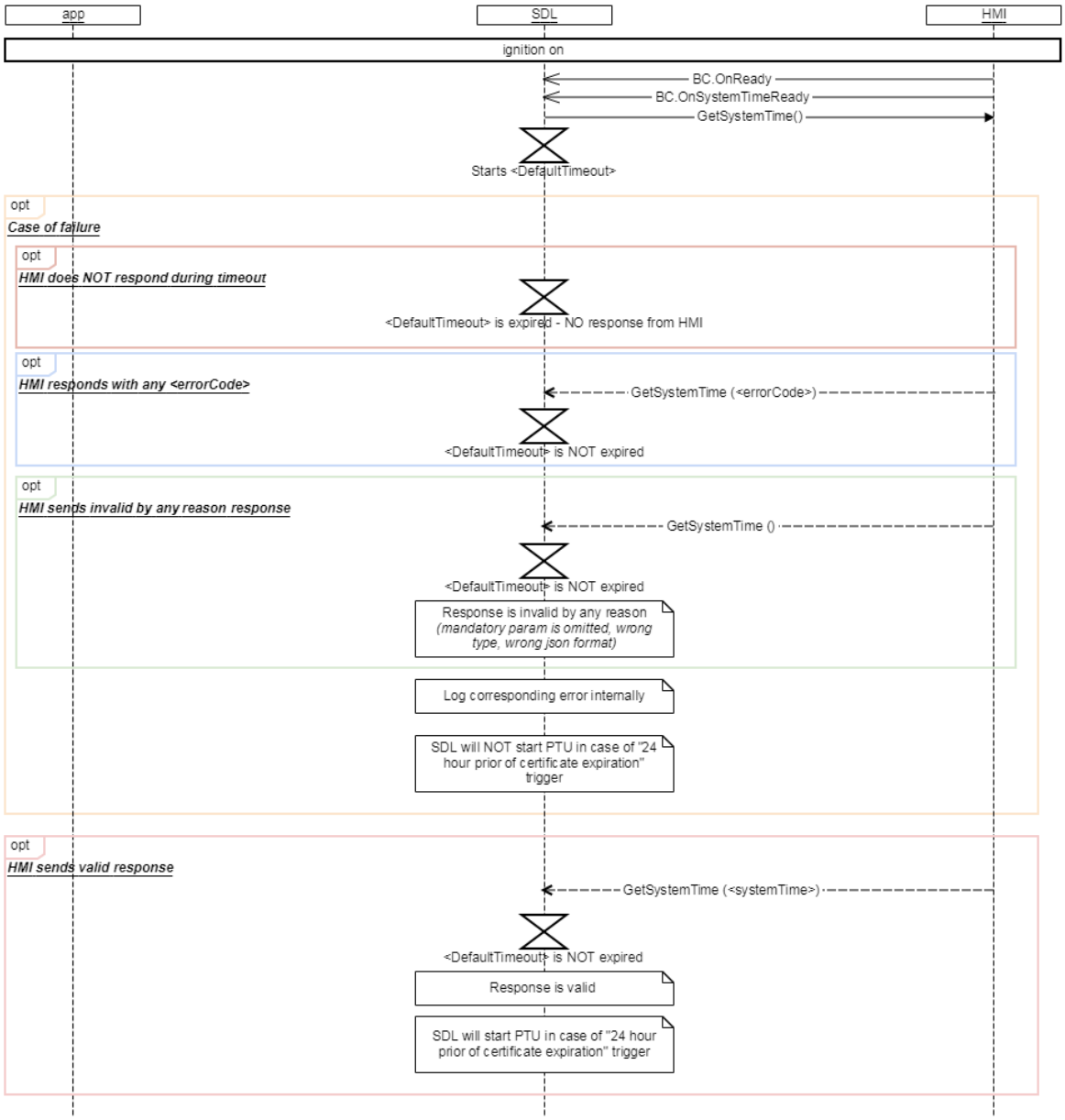
This RPC has no additional parameter requirements.

Sequence Diagrams

SEQUENCE DIAGRAM

OnSystemTimeReady

View Diagram



JSON EXAMPLE NOTIFICATION

```
{
   "jsonrpc" : "2.0",
   "method" : "BasicCommunication.OnSystemTimeReady"
}
```

GetSystemTime

Type Function
Sender SDL
Purpose Obtain current UTC time

When an application needs to start a protected service with SDL core the app sends a certificate to SDL during the TLS or DTLS handshake process.
In order to validate the certificate and to check whether this certificate is not expired SDL core needs an accurate UTC system time value.

REQUEST

SDL sends a GetSystemTime request after receiving a BC.OnSystemTimeReady notification from the HMI.
After sending the request SDL starts `<DefaultTimeout>` (value from .ini file) waiting for response from HMI.

PARAMETERS

This RPC has no additional parameter requirements.

RESPONSE

MUST

1. Send a valid response during the `<DefaultTimeout>`(value from the smartDeviceLink.ini file)
2. Provide all parameters of the "DateTime" struct inside of the GetSystemTime response.

NOTE

SDL logs the corresponding error internally and fails the handshake process if at least one of the following failures occurs:
1. HMI does NOT respond during `<DefaultTimeout>`;
2. HMI responds with any `<errorCode>` during `<DefaultTimeout>`;
3. HMI sends invalid for any reason in the response:
a. at least one String param is empty (exception: in case empty String param is allowed by HMI_API)
b. at least one String param has '\n', '\t' or completely white-space
c. wrong json format
d. at least one param has invalid type
e. at least one mandatory param was omitted
f. at least one param is out of bounds

If the handshake process fails, SDL behavior depends on the "ForceProtectedService"/"ForceUnprotectedService" params configured in the 'Security Manager' section of the smartDeviceLink.ini file.

PARAMETERS

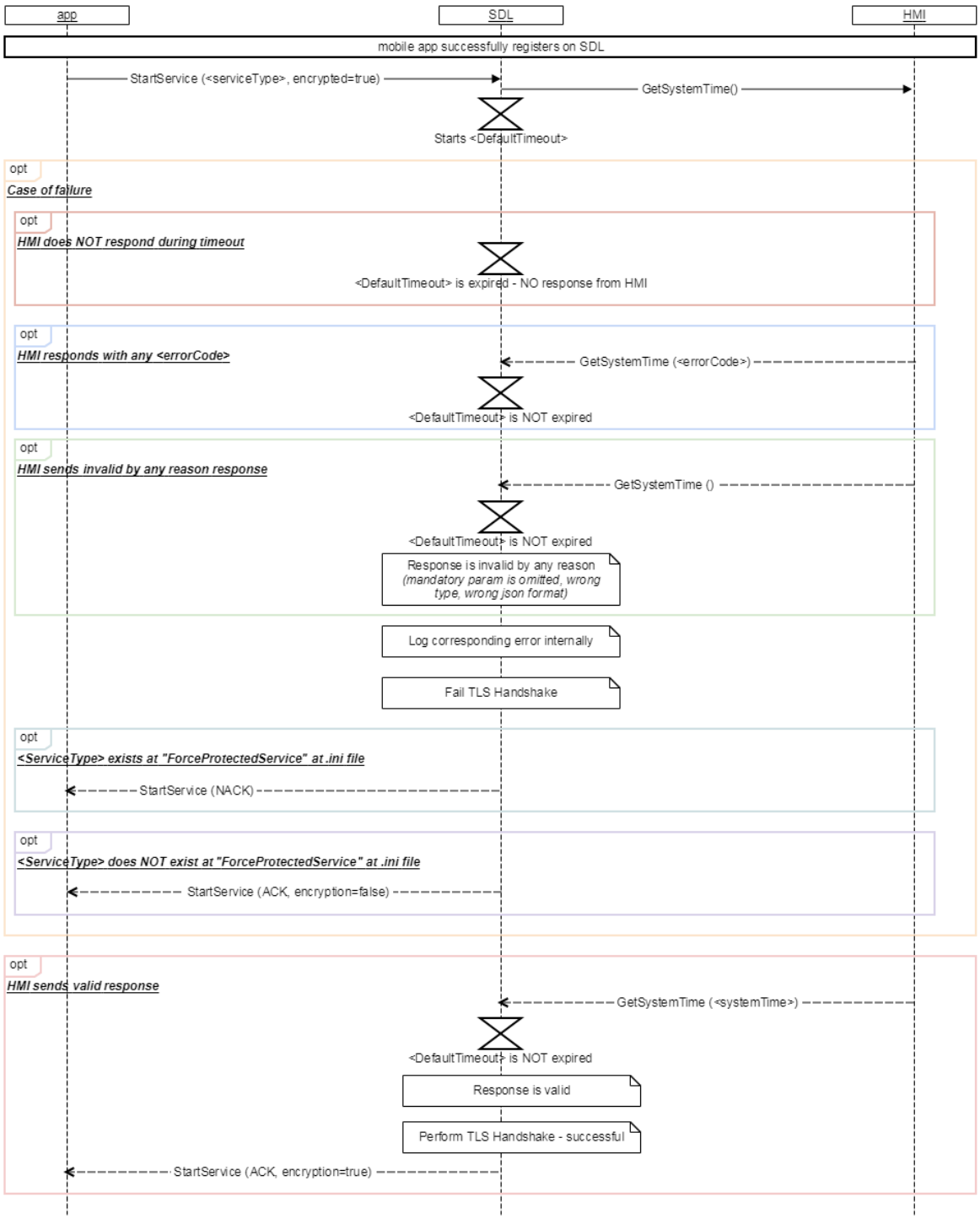
NAME	TYPE	MANDATORY	ADDITIONAL
systemTime	Common.DateTime	true	Current UTC system time

Sequence Diagrams

SEQUENCE DIAGRAM

GetSystemTime

View Diagram


app
SDL
HMI
mobile app successfully registers on SDL
StartService (<serviceType>, encrypted=true)
GetSystemTime()
Starts <DefaultTimeout>
opt
Case of failure
opt
HMI does NOT respond during timeout
<DefaultTimeout> is expired - NO response from HMI
opt
HMI responds with any <errorCode>
GetSystemTime (<errorCode>)
<DefaultTimeout> is NOT expired
opt
HMI sends invalid by any reason response
GetSystemTime ()
<DefaultTimeout> is NOT expired
Response is invalid by any reason
(mandatory param is omitted, wrong type, wrong json format)
Log corresponding error internally
Fail TLS Handshake
opt
<ServiceType> exists at "ForceProtectedService" at .ini file
StartService (NACK)
opt
<ServiceType> does NOT exist at "ForceProtectedService" at .ini file
StartService (ACK, encryption=false)
opt
HMI sends valid response
GetSystemTime (<systemTime>)
<DefaultTimeout> is NOT expired
Response is valid
Perform TLS Handshake - successful
StartService (ACK, encryption=true)

Example Request

```
{
 "id" : 59546,
 "jsonrpc" : "2.0",
 "method" : "BasicCommunication.GetSystemTime"
}
```

Example Response

```
{
  "id":59546,
  "jsonrpc":"2.0",
  "result":{
    "systemTime":[
      {
        "millisecond":11,
        "second":111,
        "minute":111,
        "hour":11,
        "day":1,
        "month":11,
        "year":2017,
        "tz_hour":1,
        "tz_minute":11
      }
    ],
    "code":0,
    "method":"BasicCommunication.GetSystemTime"
  }
}
```

Example Error


```
{
  "id":59546,
  "jsonrpc":"2.0",
  "error":{
    "code":11,
    "message":"Mandatory parameters not provided.",
    "data":{
      "method":"BasicCommunication.GetSystemTime"
    }
  }
}
```


OnSystemCapabilityUpdated

Type Notification
Sender SDL
Purpose

Inform the HMI that a specific system capability has changed

Notification

PARAMETERS

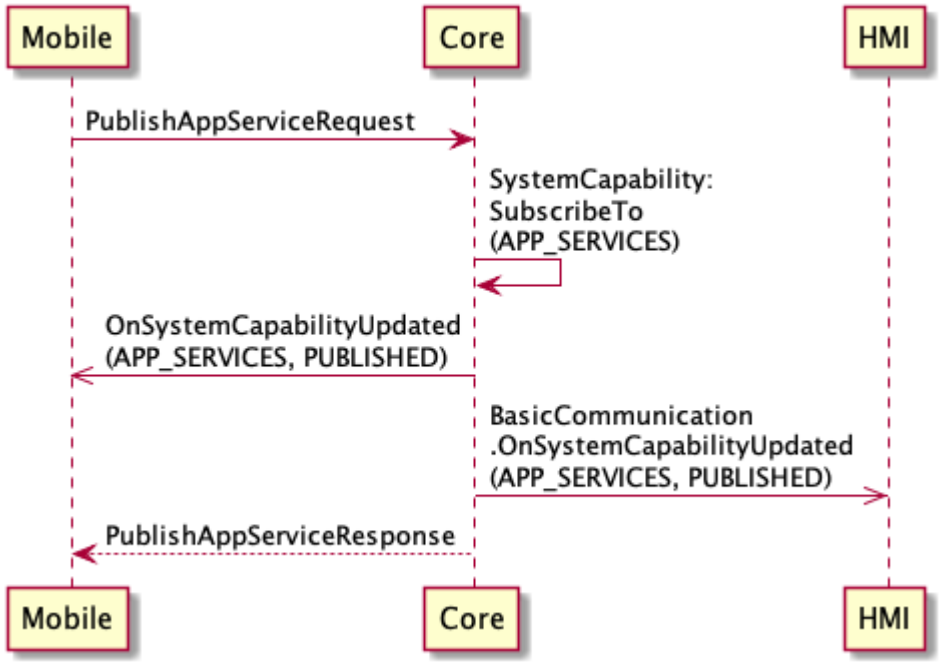
NAME	TYPE	MANDATORY	ADDITIONAL
systemCapability	Common.SystemCapability	true	

Sequence Diagrams

SEQUENCE DIAGRAM

OnSystemCapabilityUpdated(APP_SERVICES, PUBLISHED)

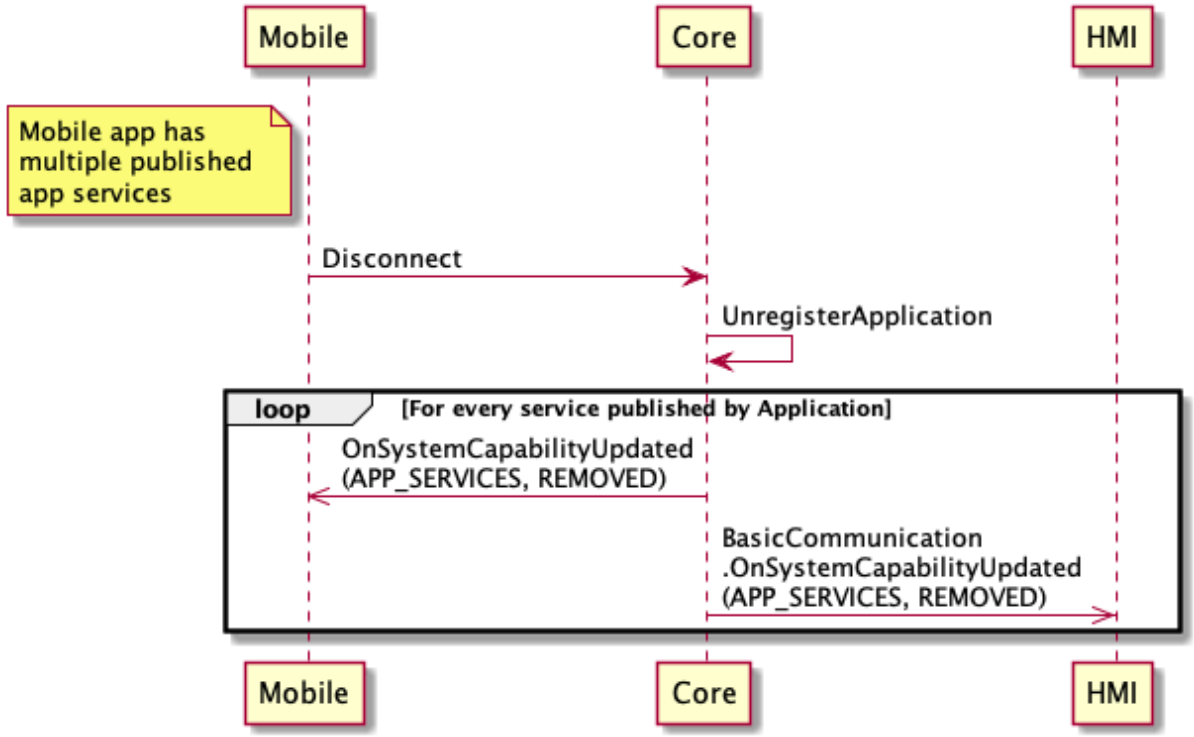
View Diagram



SEQUENCE DIAGRAM

OnSystemCapabilityUpdated(APP_SERVICES, REMOVED)

View Diagram


Mobile
Core
HMI
Mobile app has multiple published app services
Disconnect
UnregisterApplication
loop
[For every service published by Application]
OnSystemCapabilityUpdated (APP_SERVICES, REMOVED)
BasicCommunication .OnSystemCapabilityUpdated (APP_SERVICES, REMOVED)
Mobile
Core
HMI

Example Notiifcation

```
{
  "jsonrpc": "2.0",
  "method": "BasicCommunication.OnSystemCapabilityUpdated",
  "params": {
    "systemCapability": {
      "appServicesCapabilities": {
        "appServices": [
          {
            "updatedAppServiceRecord": {
              "serviceActive": true,
              "serviceID":
"c9503a4f983e3dd31a7a14564d405cdf84769c9b9a71cae9cc211a0b74
,
              "serviceManifest": {
                "allowAppConsumers": true,
                "rpcSpecVersion": {
                  "majorVersion": 5,
                  "minorVersion": 1,
                  "patchVersion": 0
                },
                "serviceName": "Mobile Media Service",
                "serviceType": "MEDIA"
              },
              "servicePublished": true
            }
          },
          {
            "updateReason": "PUBLISHED",
            "updatedAppServiceRecord": {
              "serviceActive": false,
              "serviceID":
"d6d8d2fb2bfcc00033804fb19e9fb7d6070d2c166f49881563276f17478
,
              "serviceManifest": {
                "allowAppConsumers": true,
                "rpcSpecVersion": {
                  "majorVersion": 5,
                  "minorVersion": 1,
                  "patchVersion": 0
                },
                "serviceName": "Waze Music",
                "serviceType": "MEDIA"
              },
              "servicePublished": true
            }
          }
        ]
      },
      "systemCapabilityType": "APP_SERVICES"
```

```
    }
  }
}
```

GetFilePath

Type Request
Sender SDL
Purpose

Retrieve a file path from the HMI App Service Provider (ASP)

REQUEST

PARAMETERS

NAME	TYPE	MANDATORY	ADDITIONAL
fileName	String	true	maxlength: 255
fileType	Common.FileType	false	
appServiceId	String	false	

RESPONSE

PARAMETERS

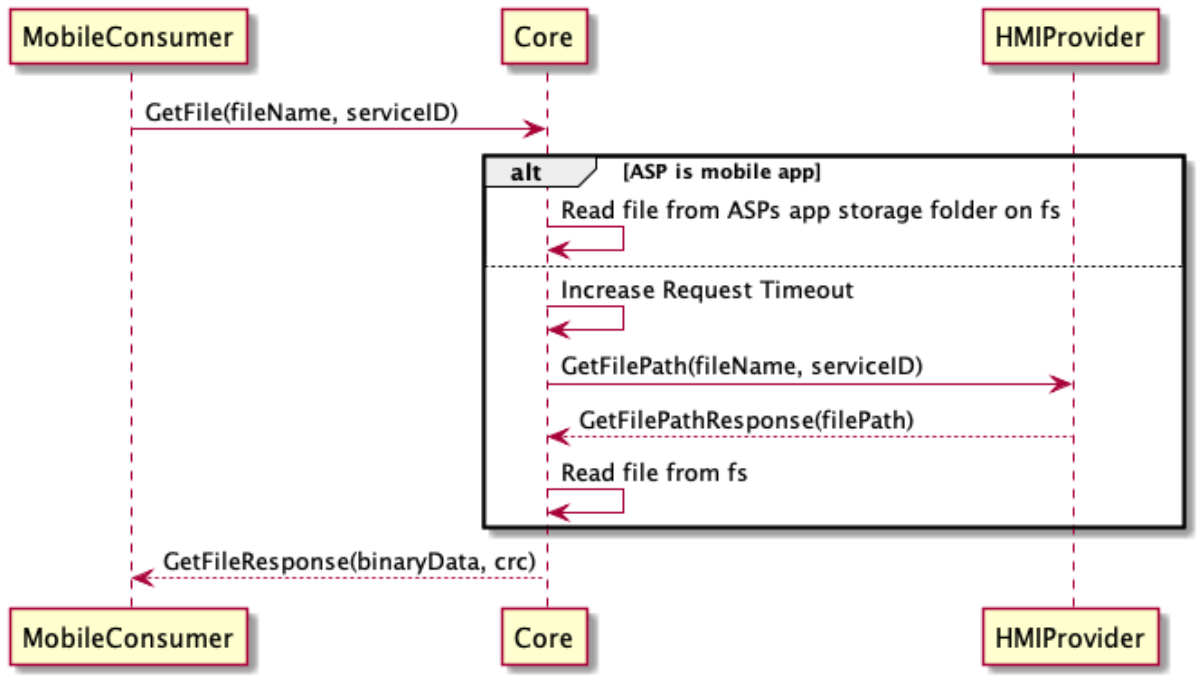
NAME	TYPE	MANDATORY	ADDITIONAL
filePath	String	false	
fileType	Common.FileType	false	

Sequence Diagrams

SEQUENCE DIAGRAM

GetFile (HMI Provider)

View Diagram



Example Request

```
{
  "id": 34,
  "jsonrpc": "2.0",
  "method": "BasicCommunication.GetFilePath",
  "params": {
    "appServiceId":
"d712bcaced1260e90f1acf95522dc0ef763bc72f234bcf7672cd7e23801c
",
    "fileName": "somefile.json",
    "fileType": "JSON"
  }
}
```

Example Response

```
{
    "id": 34,
    "jsonrpc": "2.0",
    "result": {
        "method": "BasicCommunication.GetFilePath",
        "code": 0,
        "fileType": "JSON",
        "filePath": "/home/user/HMI/somefile.json"
    }
}
```

Example Error

```
{
 "id" : 34,
 "jsonrpc" : "2.0",
 "error" : {
  "code" : 27,
  "message" : "File not found",
  "data" : {
   "method" : "BasicCommunication.GetFilePath"
  }
 }
}
```

GetCapabilities

Type Function
Sender SDL
Purpose Get the capabilities of buttons in the vehicle.

REQUEST

PARAMETERS

NAME	TYPE	MANDATORY	ADDITIONAL

RESPONSE

PARAMETERS

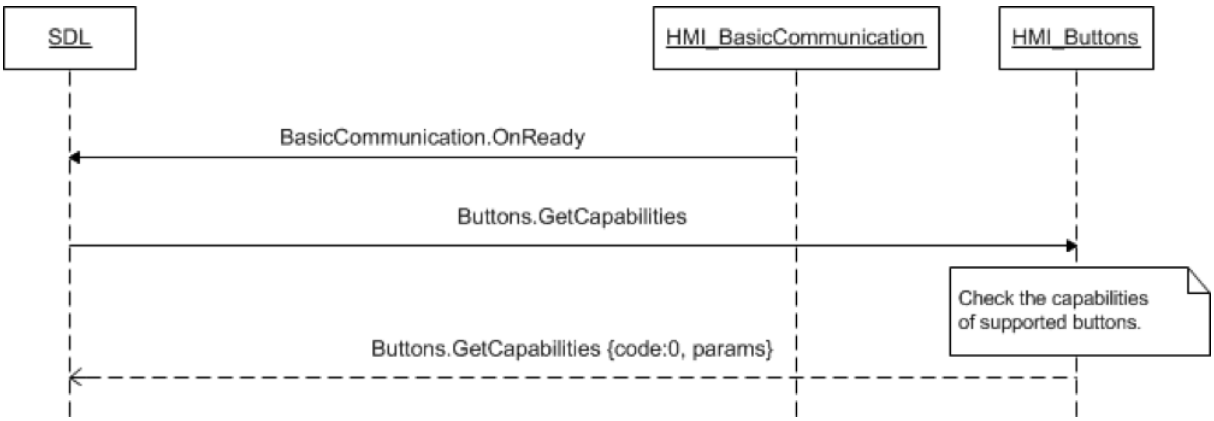
NAME	TYPE	MANDATORY	ADDITIONAL
capabilities	Common.ButtonCapabilities	true	array: true minsize: 1 maxsize: 100
presetBankCapabilities	Common.PresetBankCapabilities	false	

Sequence Diagrams

SEQUENCE DIAGRAM

GetCapabilities on system startup

View Diagram



Example Request

```
{
  "id" : 20,
  "jsonrpc" : "2.0",
  "method" : "Buttons.GetCapabilities"
}
```

Example Response


```
{
 "id" : 20,
 "jsonrpc" : "2.0",
 "result" :
 {
  "capabilities" :
  [
   {
    "name" : OK,
    "shortPressAvailable" : true,
    "longPressAvailable" : true,
    "upDownAvailable" : true
   },
   {
    "name" : SEEKLEFT,
    "shortPressAvailable" : true,
    "longPressAvailable" : true,
    "upDownAvailable" : true
   },
   {
    "name" : SEEKRIGHT,
    "shortPressAvailable" : true,
    "longPressAvailable" : true,
    "upDownAvailable" : true
   },
   {
    "name" : TUNEUP,
    "shortPressAvailable" : true,
    "longPressAvailable" : true,
    "upDownAvailable" : true
   },
   {
    "name" : TUNEDOWN,
    "shortPressAvailable" : true,
    "longPressAvailable" : true,
    "upDownAvailable" : true
   },
  ],
  "presetBankCapabilities" :
  [
   "onScreenPresetsAvailable" : true
  ],
  "code" : 0,
  "method" : "Buttons.GetCapabilities"
 }
}
```

Example Error

```
{
 "id" : 20,
 "jsonrpc" : "2.0",
 "error" :
 {
  "code" : 9,
  "message" : "The requested data is not available",
  "data" :
  {
   "method" : "Buttons.GetCapabilities"
  }
 }
}
```

OnButtonEvent

Type Notification
Sender HMI
Purpose Inform SDL about the occurrence of a button event.

Notification

PARAMETERS

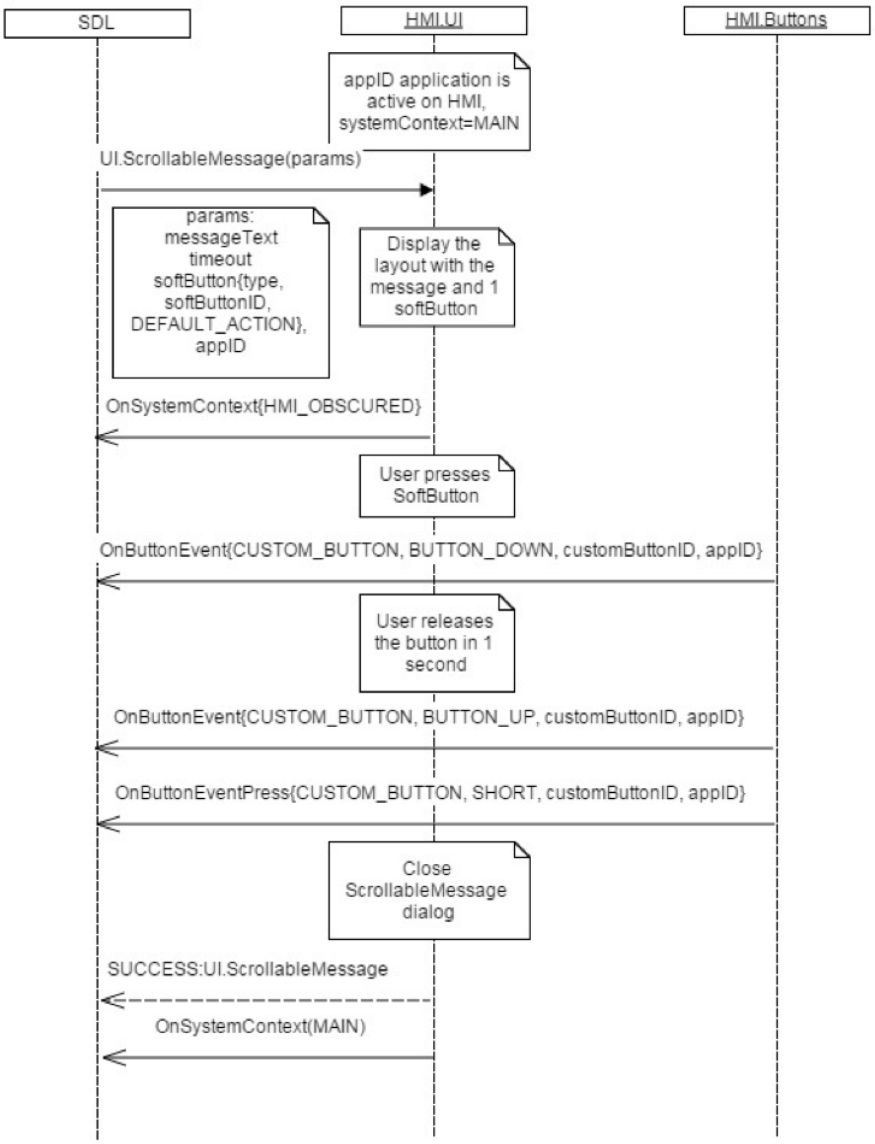
NAME	TYPE	MANDATORY	ADDITIONAL
name	Common.ButtonName	true	
mode	Common.ButtonEventMode	true	
customButtonID	Integer	false	minvalue: 0 maxvalue: 65536
appID	Integer	false	

Sequence Diagrams

SEQUENCE DIAGRAM

OnButtonEvent for CUSTOM_BUTTON pressed and released

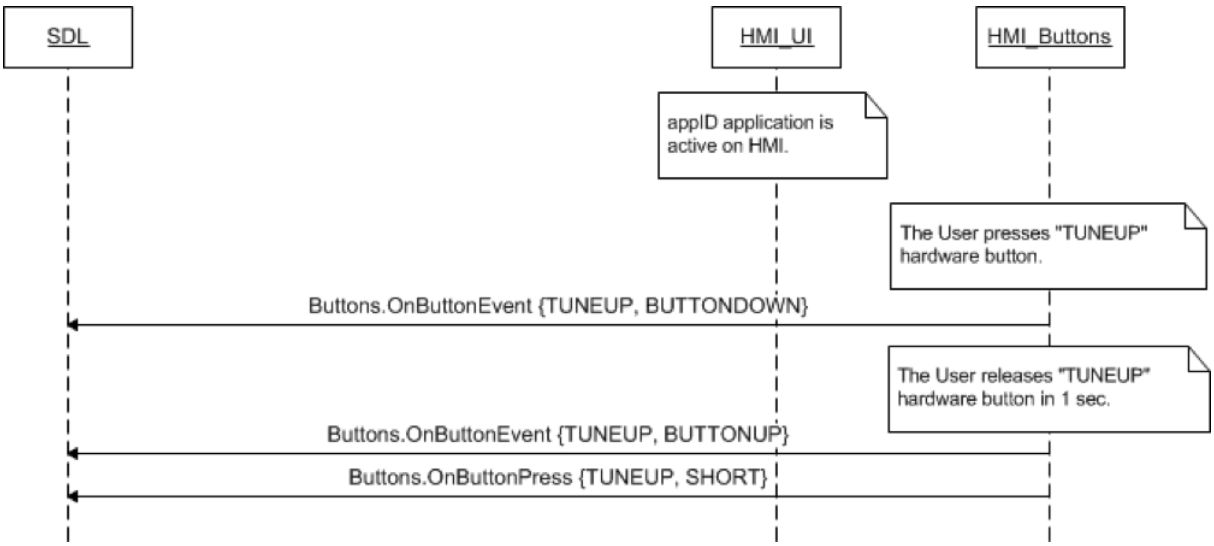
View Diagram



SEQUENCE DIAGRAM

OnButtonEvent hardware button pressed and released

View Diagram



JSON EXAMPLE NOTIFICATION

```
{
 "jsonrpc" : "2.0",
 "method" : "Buttons.OnButtonEvent",
 "params" :
 {
  "name" : OK,
  "mode" : BUTTONDOWN,
 }
}
```

OnButtonPress

Type

Notification

Sender

HMI

Purpose

Inform SDL about a Button Press

If the HMI reports to SDL via Buttons.GetCapabilities that it supports long and/or short button press modes, SDL expects the HMI to send the Buttons.OnButtonPress notification but buttons that have been subscribed via Buttons.OnButtonSubscription and custom buttons added in other rpcs as Soft Buttons.

MUST

The hmi must send the name of the button pressed, the press mode detected, and ID of the button if it has type `CUSTOM_BUTTON` and the appID related to the button press. If only `SHORT` button press mode is supported, the hmi should send `SHORT` regardless of the time of the button press.

NOTE

The value of customButtonID is provided by SDL within the softButton struct for some rpcs such as UI.Alert

Notification

PARAMETERS

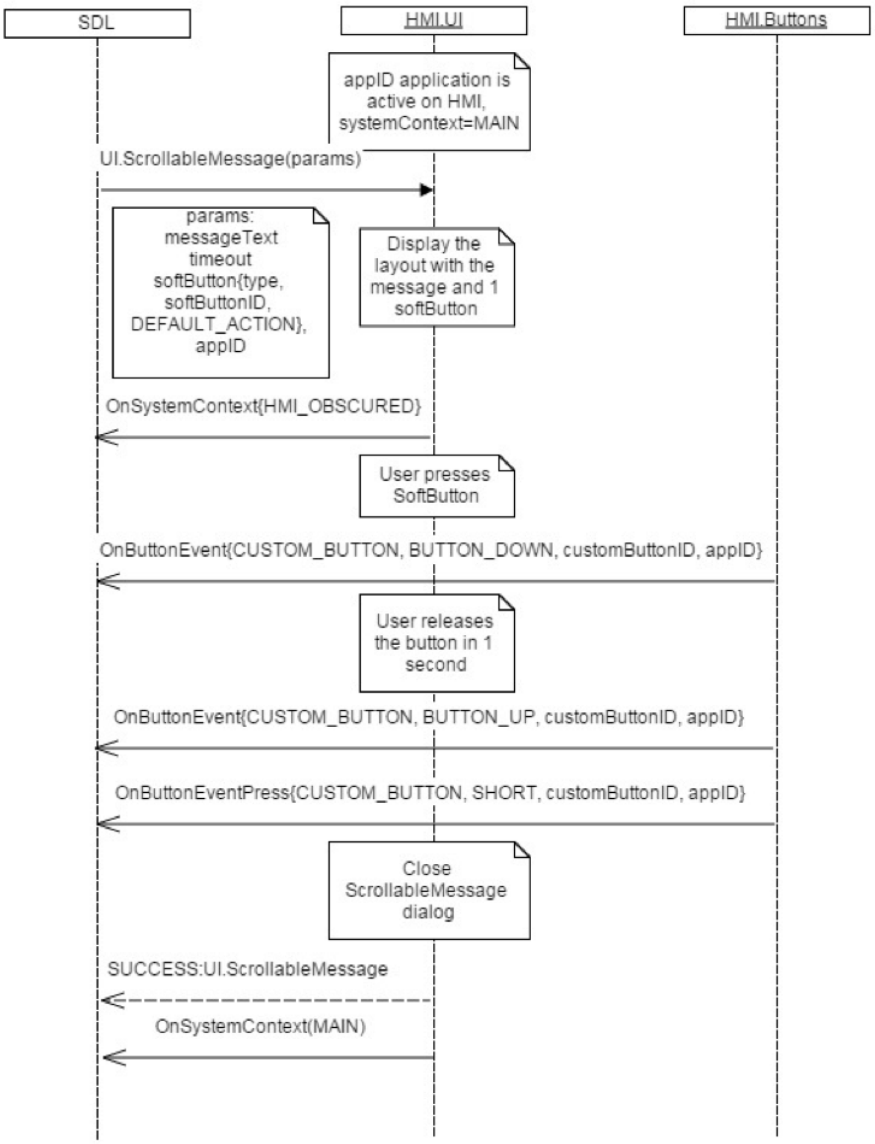
NAME	TYPE	MANDATORY	ADDITIONAL
name	Common.ButtonName	true	
mode	Common.ButtonPressMode	true	
customButtonID	Integer	false	minvalue: 0 maxvalue: 65536
appID	Integer	false	

Sequence Diagrams

SEQUENCE DIAGRAM

OnButtonPress short press for CUSTOM_BUTTON

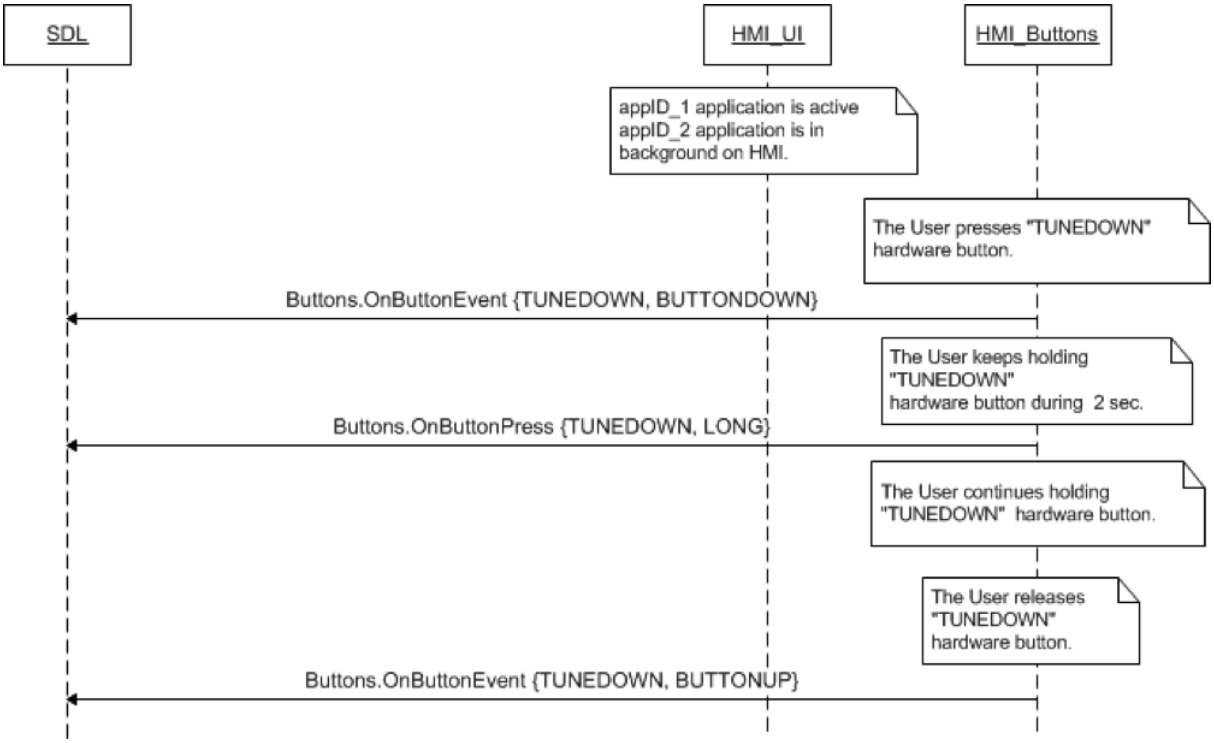
View Diagram



SEQUENCE DIAGRAM

OnButtonPress long press for hard button

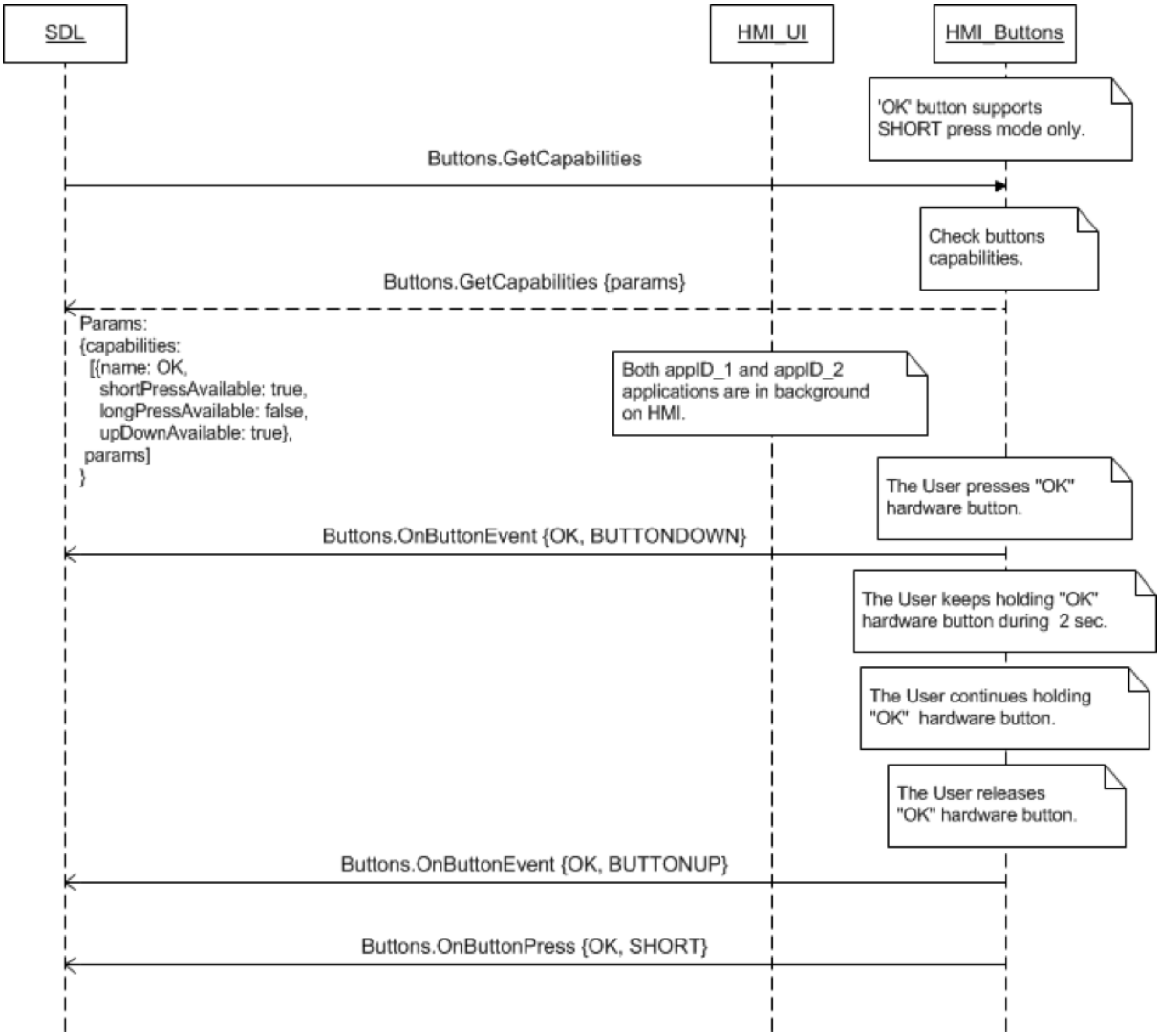
View Diagram



SEQUENCE DIAGRAM

OnButtonPress for hard button that only supports short press

View Diagram



JSON EXAMPLE NOTIFICATION

```
{
 "jsonrpc" : "2.0",
 "method" : "Buttons.OnButtonPress",
 "params" :
 {
  "name" : "CUSTOM_BUTTON",
  "mode" : "SHORT",
  "customButtonID" : 564
 }
}
```

OnButtonSubscription

Type

Notification

Sender

SDL

Purpose

Notify the HMI that the specified application's interest in receiving notifications for a button has changed.

Notification

PARAMETERS

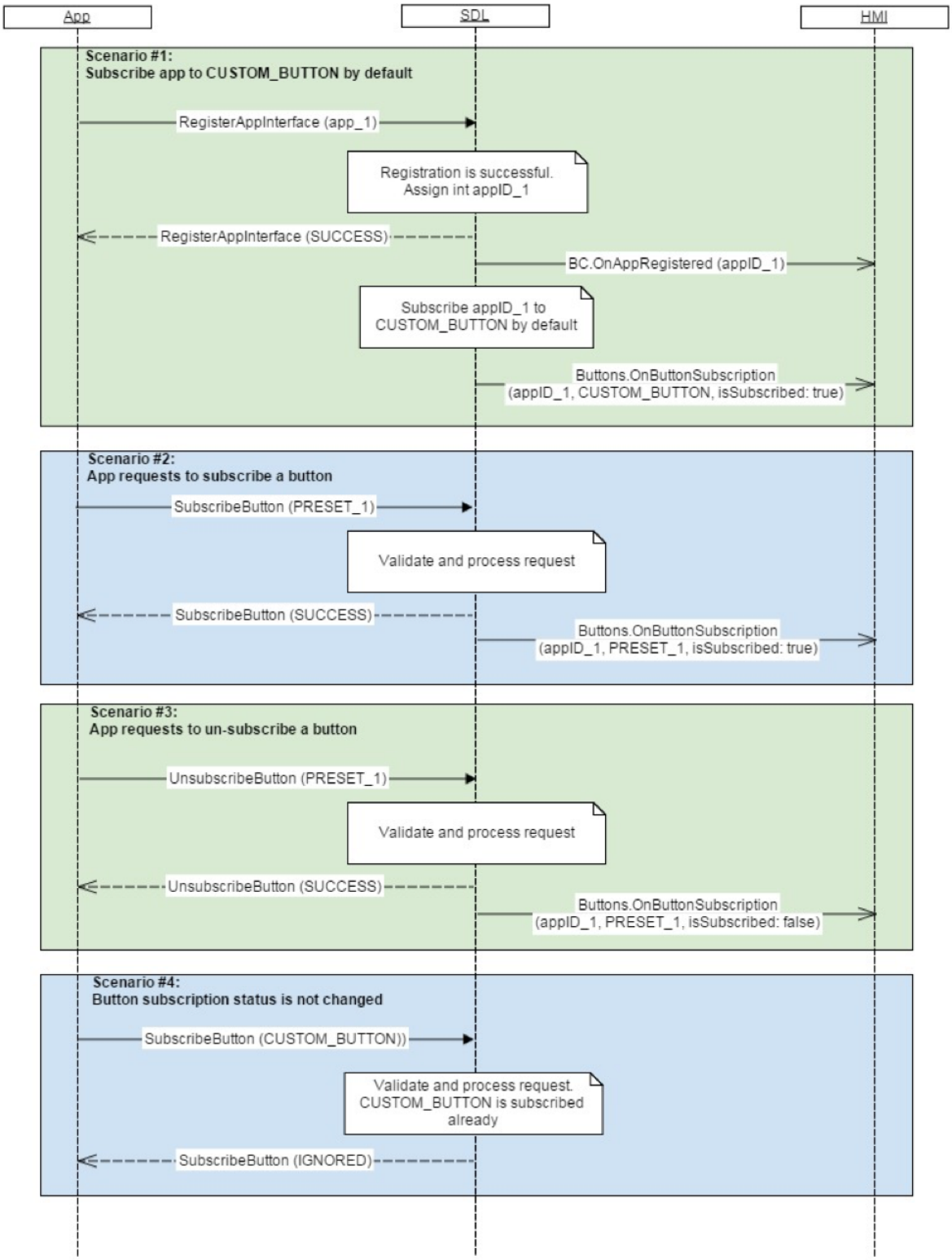
NAME	TYPE	MANDATORY	ADDITIONAL
name	Common.ButtonName	true	
isSubscribed	Boolean	true	
appID	Integer	true	

Sequence Diagrams

SEQUENCE DIAGRAM

OnButtonSubscription

View Diagram


App
SDL
HMI
Scenario #1:
Subscribe app to CUSTOM_BUTTON by default
RegisterAppInterface (app_1)
Registration is successful.
Assign int appID_1
RegisterAppInterface (SUCCESS)
BC.OnAppRegistered (appID_1)
Subscribe appID_1 to
CUSTOM_BUTTON by default
Buttons.OnButtonSubscription
(appID_1, CUSTOM_BUTTON, isSubscribed: true)
Scenario #2:
App requests to subscribe a button
SubscribeButton (PRESET_1)
Validate and process request
SubscribeButton (SUCCESS)
Buttons.OnButtonSubscription
(appID_1, PRESET_1, isSubscribed: true)
Scenario #3:
App requests to un-subscribe a button
UnsubscribeButton (PRESET_1)
Validate and process request
UnsubscribeButton (SUCCESS)
Buttons.OnButtonSubscription
(appID_1, PRESET_1, isSubscribed: false)
Scenario #4:
Button subscription status is not changed
SubscribeButton (CUSTOM_BUTTON))
Validate and process request.
CUSTOM_BUTTON is subscribed
already
SubscribeButton (IGNORED)

JSON EXAMPLE NOTIFICATION

```
{
 "jsonrpc" : "2.0",
 "method" : "Buttons.OnButtonSubscription",
 "params" :
 {
  "name" : "SEEKLEFT",
  "isSubscribed" : true,
  "appID" : 564
 }
}
```

ButtonPress

Type Function
Sender SDL
Purpose Emulate button press event on HMI for the common climate or radio control buttons in vehicle

ButtonPress represents a request from an application to change settings of requested RC module by pressing the appropriate button.
This RPC can be sent to the HMI from an application that is registered with REMOTE_CONTROL appHMIType and in one of the following states: FULL, LIMITTED, BACKGROUND.
Module signed by the application in such request has to be available on HMI and allowed for control change settings.
The system shall list all available RC radio buttons and RC climate buttons in the existing ButtonCapabilities list.

MUST

1. Modules sent by application must be available on HMI
2. Access to control module settings is defined by access mode entered by user on HMI

3. Requested control buttons have to be available for such module on HMI

REQUEST

PARAMETERS

NAME	TYPE	MANDATORY	ADDITIONAL	DESCRIPTION
moduleType	Common.ModuleType	true		The module where the button should be pressed
buttonName	Common.ButtonName	true		
buttonPressMode	Common.ButtonPressMode	true		Indicates whether this is a LONG or SHORT button press event.
appID	Integer	true		Internal SDL-assigned ID of the related application

RESPONSE

PARAMETERS

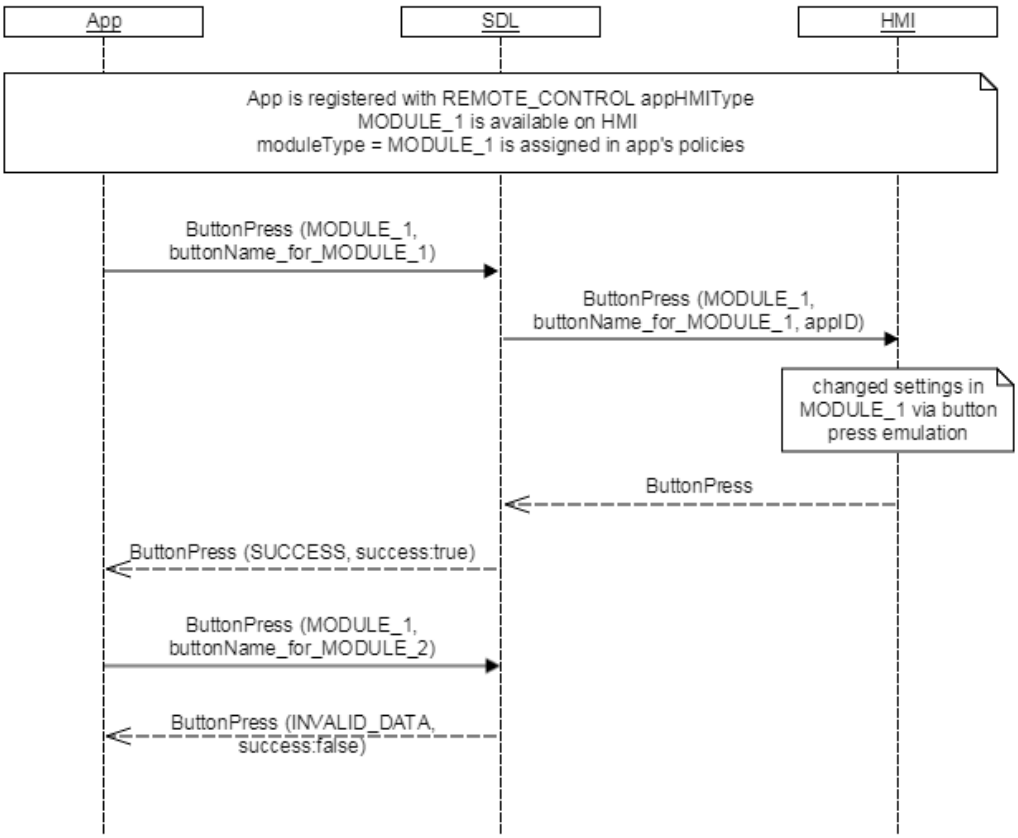
This RPC has no additional parameter requirements

Sequence Diagrams

SEQUENCE DIAGRAM

ButtonPress

View Diagram



Example Request

```
{
   "id": 32,
   "jsonrpc": "2.0",
   "method": "Buttons.ButtonPress",
   "params": {
      "appID": 680015438,
      "buttonName": "VOLUME_UP",
      "buttonPressMode": "LONG",
      "moduleType": "RADIO"
   }
}
```

Example Response

```
{
   "id": 32,
   "jsonrpc": "2.0",
   "result": {
      "code": 0,
      "method": "Buttons.ButtonPress"
   }
}
```

Example Error

```
{
 "id" : 32,
 "jsonrpc" : "2.0",
 "error" :
 {
  "code" : 22,
  "message" : "An unknown issue occurred ",
  "data" :
  {
   "method" : "Buttons.ButtonPress"
  }
 }
}
```

AlertManeuver

Type

Function
Sender SDL
Purpose Announce a navigation maneuver.

SDL sends `AlertManeuver` together with TTS.Speak. The purpose of this RPC is to notify the embedded navigation system about the next navigation maneuver.

MAY

1. Notify the user by TTS.Speak, which is sent along with `AlertManeuver`.
2. Display the `AlertManeuver` dialog with an alert icon and one of the following:
   - Up to three soft buttons defined within the `softButtons` paremeter.
   - HMI-defined `Close` soft button if a request without `softButtons` is received.
3. Process `SystemContext` behavior for `AlertManeuver` in the same way an `Alert` is handled.

MUST

UI.OnSystemContext `appID` rules if `AlertManeuver` causes `Syste mContext` updates:

1. `appID` should not be sent for `MENU` and `HMI_OBSCURED` system contexts. Only apps in `FULL` should receive these updates.
2. If the HMI sends `appID` with UI.OnSystemContext(`MENU` or `HMI_OBSCURED`), `appID` is ignored and the notification is transferred to the application that is in `FULL`.
3. UI.OnSystemContext's `appID` parameter should be mandatory for `MAIN` and `ALERT` values. If no `appID` is sent for this case, the notification will be ignored by SDL.

NOTE

If the HMI does not respond to SDL's request, after the default timeout occurs SDL will send `GENERIC_ERROR` to the corresponding mobile application.

REQUEST

PARAMETERS

NAME	TYPE	MANDATORY	ADDITIONAL
softButtons	Common.SoftButton	false	array: true minsize: 0 maxsize: 3
appID	Integer	true	

RESPONSE

PARAMETERS

This RPC has no additional parameter requirements

Sequence Diagrams

SEQUENCE DIAGRAM

AlertManeuver Default Success Path

View Diagram

SEQUENCE DIAGRAM

AlertManeuver from Background

View Diagram

SEQUENCE DIAGRAM

AlertManeuver Aborted

View Diagram

SEQUENCE DIAGRAM

AlertManeuver Rejected

View Diagram

Example Request

```
{
 "id": 143,
 "jsonrpc": "2.0",
 "method": "Navigation.AlertManeuver",
 "params": {
  "softButtons": [
   {
    "type": TEXT,
    "text": "Leave Onscreen",
    "softButtonID": 45,
    "systemAction": KEEP_CONTEXT
   },
   {
    "type": TEXT,
    "text": "Close",
    "softButtonID": 46,
    "systemAction": STEAL_FOCUS
   }
  ]
 },
 "appID": 96
}
```

Example Response

```
{
 "id" : 143,
 "jsonrpc" : "2.0",
 "result" :
 {
  "code" : 0,
  "method" : "Navigation.AlertManeuver"
 }
}
```

Example Error

```
{
 "id" : 143,
 "jsonrpc" : "2.0",
 "error" :
 {
  "code" : 13,
  "message" : "The command cannot be executed because there is
NO app registered with the specified appID",
  "data" :
  {
   "method" : "Navigation.AlertManeuver"
  }
 }
}
```

GetWayPoints

Type Function
Sender SDL
Purpose Request for getting waypoint/destination data.

SDL forwards a request from the mobile application to get waypoint/destination data from the embedded navigation system.

REQUEST

MUST

1. Accept requests for getting details of active destination and waypoints and provide details in response to the request.
2. Send UI.OnResetTimeout to SDL in case HMI needs more time for processing `GetWayPoints` request.

SDL Note: In case HMI does not respond to this request within SDL's default timeout (10s by default, UI.OnResetTimeout will reset this), SDL will return `GENERIC_ERROR` code to the corresponding mobile app's request

PARAMETERS

NAME	TYPE	MANDATORY	ADDITIONAL	DESCRIPTION
wayPointType	Common.WayPointType	true		To request for either the destination only or for all waypoints including destination
appID	Integer	true		ID of the application that concerns this RPC

RESPONSE

MUST

1. Wait using a predefined timeout for the navigation system to respond with the full data. The system shall wait for data (resetting the timer as necessary with UI.OnResetTimeout) until all waypoints have been received.
2. **If additional requests are received:** While a request for getting waypoints is being processed, if HMI receives another request from the same app then the system shall reject the new request with a response of `IN_USE`.

3. **If there is no active navigation source:** the system shall provide a response of `UNSUPPORTED_RESOURCE`.
4. **If the predefined timeout expires** *or* **the system receives a "time out" notification from the navigation system:** the system shall provide a response of `TIMED_OUT`.
   a) The system shall ignore and discard any data received from the navigation system after the timeout period.
5. If the system receives any kind of failure notification other than "time out" from the navigation system, then the system shall provide a response of `REJECTED`.
6. When the system receives all the data from the navigation system within the timeout period, the system shall send provide a response of `SUCCESS` with the received data.
7. The array of waypoints shall be ordered in this manner:
   a) The destination will be set as the first entry in the array
   b) **If "wayPointType" is "ALL":** The remaining entries will be in order based on the route, with the first waypoint being the second entry in the array, the second waypoint being the third entry, and so on.
8. If there is no active route set then the system send a response of SUCCESS with no `wayPoints` value.

PARAMETERS

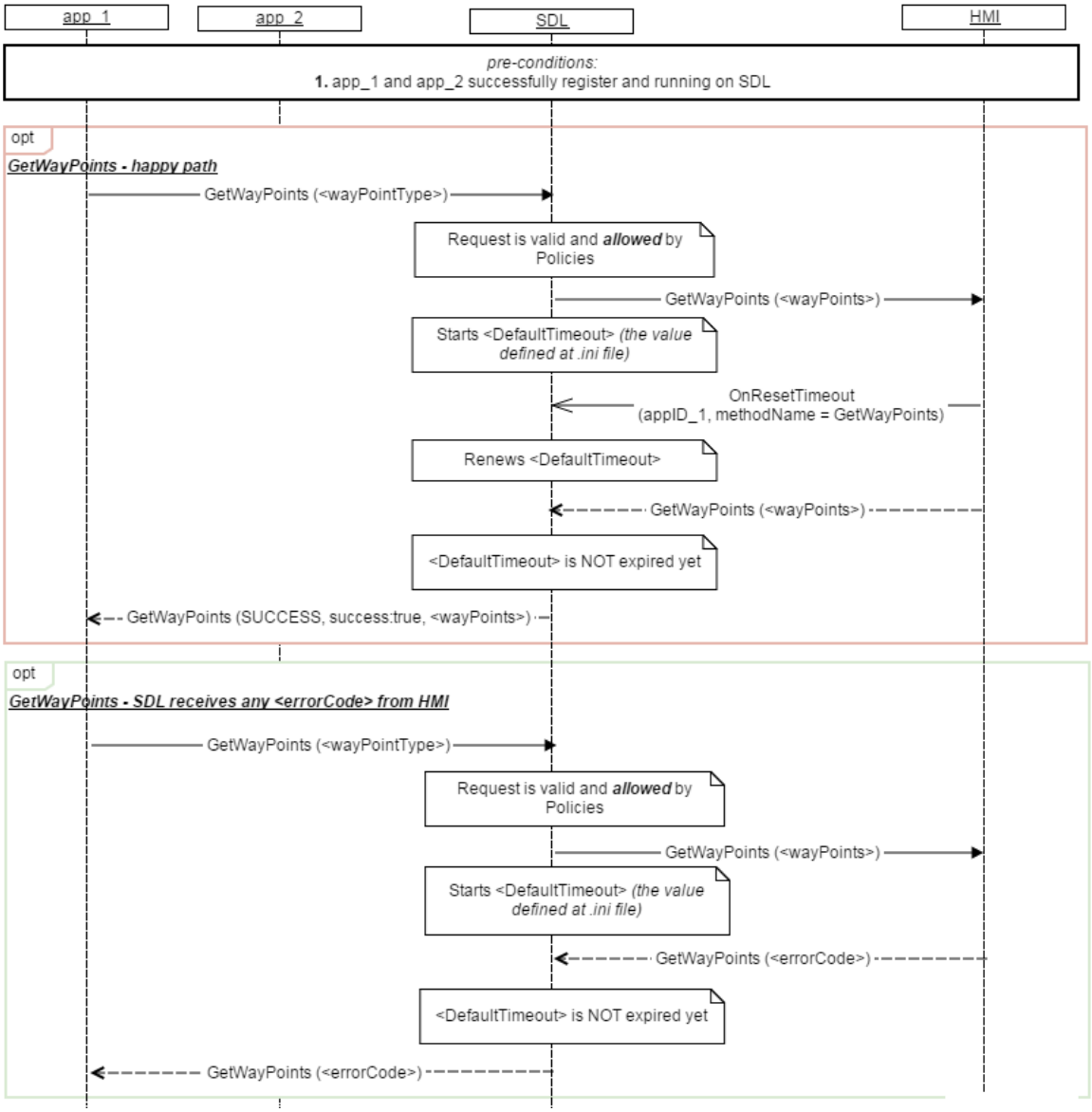
NAME	TYPE	MANDATORY	ADDITIONAL	DESCRIPTION
wayPoints	Common.LocationDetails	false	array: true minsize: 1 maxsize: 10	

Sequence Diagrams

SEQUENCE DIAGRAM

GetWayPoints

View Diagram



JSON Messages Examples

Example Request

```
{
   "id" : 543,
   "jsonrpc" : "2.0",
   "method" : "Navigation.GetWayPoints",
   "params" :
   {
      "wayPointType" : "ALL",
      "appID" : 26743
   }
}
```

Example Response

```
{
   "id" : 543,
   "jsonrpc" : "2.0",
   "method" : "Navigation.GetWayPoints",
   "result" :
   {
      "wayPoints" :
      [
         {
            "phoneNumber" :  "123-456-7890",
            "addressLines" : "addresstext"
         }
      ],
      "appID" : 26743
   }
}
```

Example Error


```
{
   "id" : 543,
   "jsonrpc" : "2.0",
   "error" :
   {
      "code" : 11,
      "message" : "The data sent is invalid",
      "data" :
      {
         "method" : "Navigation.GetWaypoints"
      }
   }
}
```


SubscribeWayPoints

Type Function
Sender SDL
Purpose Subscribe to periodic navigation waypoint/destination updates

REQUEST

MUST

Send future waypoint updates to SDL using Navigation.OnWayPointChange

PARAMETERS

This RPC has no additional parameter requirements

RESPONSE

PARAMETERS

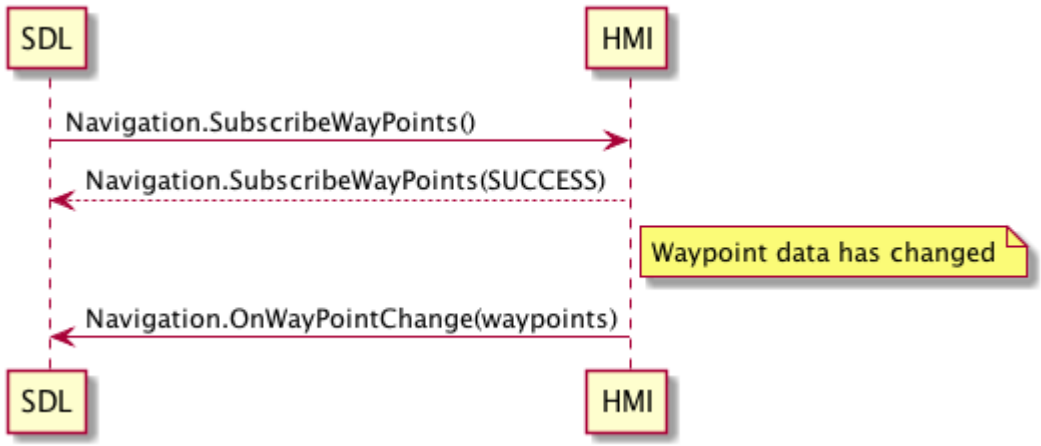
This RPC has no additional parameter requirements

Sequence Diagrams

SEQUENCE DIAGRAM

SubscribeWayPoints

View Diagram



Example Request

```
{
 "id" : 47,
 "jsonrpc" : "2.0",
 "method" : "Navigation.SubscribeWayPoints",
 "params" : {}
}
```

Example Response

```
{
 "id" : 47,
 "jsonrpc" : "2.0",
 "result" :
 {
  "code" : 0,
  "method" : "Navigation.SubscribeWayPoints"
 }
}
```

Example Error


```
{
 "id" : 47,
 "jsonrpc" : "2.0",
 "error" :
 {
  "code" : 6,
  "message" : "Already subscribed to waypoints",
  "data" :
  {
   "method" : "Navigation.SubscribeWayPoints"
  }
 }
}
```


UnsubscribeWayPoints

Type Function
Sender SDL
Purpose Unsubscribe from periodic navigation waypoint/destination updates

REQUEST

MUST

Stop sending waypoint updates to SDL

PARAMETERS

This RPC has no additional parameter requirements

RESPONSE

PARAMETERS

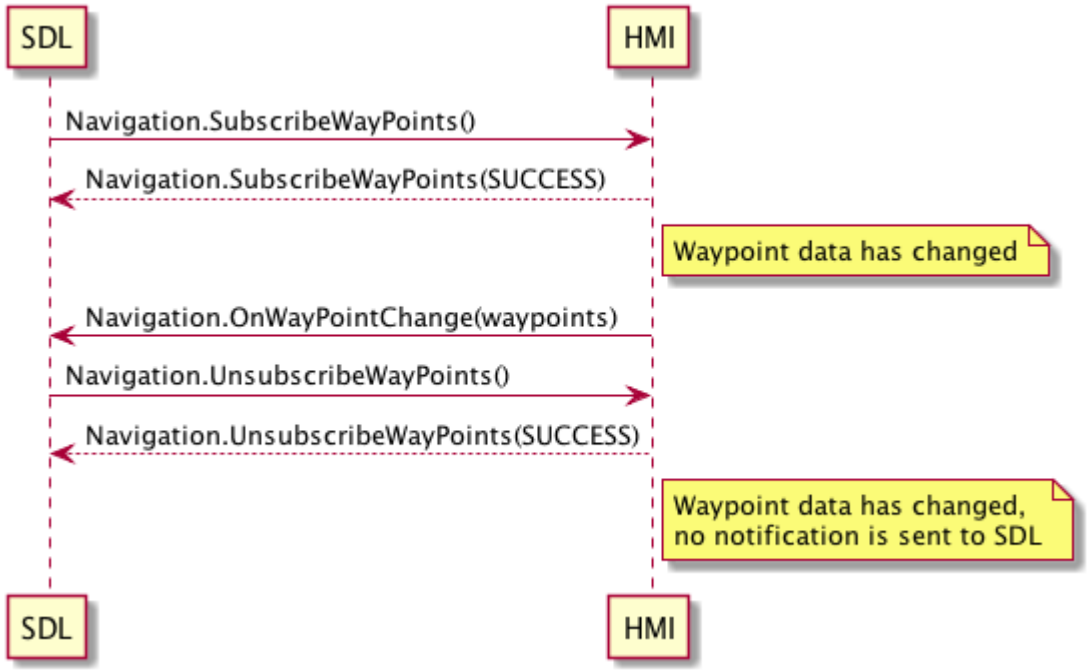
This RPC has no additional parameter requirements

Sequence Diagrams

SEQUENCE DIAGRAM

UnsubscribeWayPoints

View Diagram



Example Request

```
{
 "id" : 47,
 "jsonrpc" : "2.0",
 "method" : "Navigation.UnsubscribeWayPoints",
 "params" : {}
}
```

Example Response

```
{
 "id" : 47,
 "jsonrpc" : "2.0",
 "result" :
 {
  "code" : 0,
  "method" : "Navigation.UnsubscribeWayPoints"
 }
}
```

Example Error

```
{
 "id" : 47,
 "jsonrpc" : "2.0",
 "error" :
 {
  "code" : 6,
  "message" : "Not subscribed to waypoints",
  "data" :
  {
   "method" : "Navigation.UnsubscribeWayPoints"
  }
 }
}
```

OnWayPointChange

Type Notification
Sender HMI
Purpose Notify SDL of navigation waypoint/destination updates

Notification

MUST

Send this notification when the list of waypoints is changed and SDL is subscribed to waypoint updates

PARAMETERS

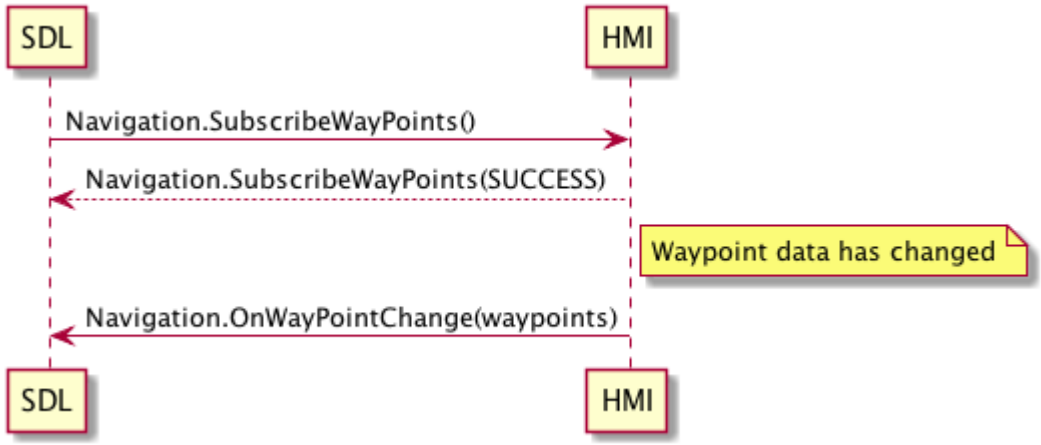
NAME	TYPE	MANDATORY	ADDITIONAL	DESCRIPTION
wayPoints	Common.LocationDetails	true	array: true minsize: 1 maxsize: 10	

Sequence Diagrams

SEQUENCE DIAGRAM

OnWayPointChange

View Diagram



Example Notification

```
{
 "id" : 47,
 "jsonrpc" : "2.0",
 "method" : "Navigation.OnWayPointChange",
 "params" : {
  "wayPoints" :
  [
   {
    "phoneNumber" :  "123-456-7890",
    "addressLines" : "addresstext"
   }
  ]
 }
}
```

IsReady

Type

Function

Sender

SDL

Purpose

Request ready state of Navigation Module

This request comes from SDL after the HMI has confirmed its readiness via OnReady.

NOTE

If the Navigation component responds as `unavailable`, SDL will not send further requests to this component.

MUST

1. Check if the Navigation component is present and ready to communicate with SDL.

2. Respond correspondingly to the results of this check.

REQUEST

PARAMETERS

NAME	TYPE	MANDATORY	ADDITIONAL

RESPONSE

PARAMETERS

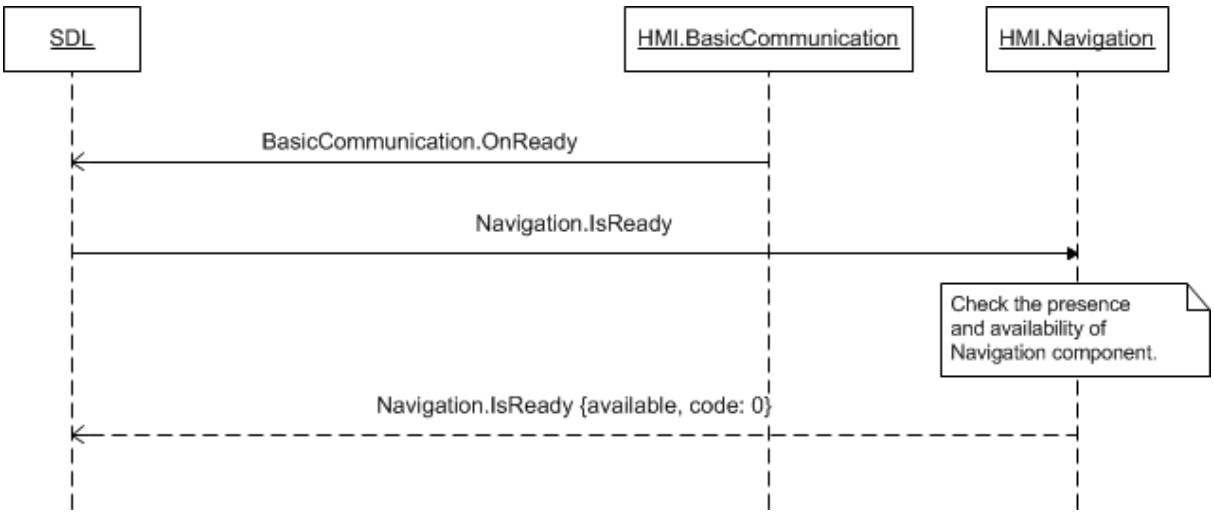
NAME	TYPE	MANDATORY	ADDITIONAL
available	Boolean	true	

Sequence Diagrams

SEQUENCE DIAGRAM

IsReady

View Diagram



Example Request

```
{
 "id" : 27,
 "jsonrpc" : "2.0",
 "method" : "Navigation.IsReady"
}
```

Example Response

```
{
 "id" : 27,
 "jsonrpc" : "2.0",
 "result" :
 {
  "availabe" : true,
  "code" : 0,
  "method" : "Navigation.IsReady"
 }
}
```

Example Error

```
{
 "id" : 27,
 "jsonrpc" : "2.0",
 "error" :
 {
  "code" : 22,
  "message" : "An unknown error has occurred",
  "data" :
  {
   "method" : "Navigation.IsReady"
  }
 }
}
```

OnAudioDataStreaming

Type Notification
Sender SDL
Purpose Notify the HMI about a raw audio data stream which should be audible to the user.

Notification

PARAMETERS

NAME	TYPE	MANDATORY	ADDITIONAL
available	Boolean	true	

Sequence Diagrams

SEQUENCE DIAGRAM

OnAudioDataStreaming SDL to HMI

View Diagram

SEQUENCE DIAGRAM

OnAudioDataStreaming App to HMI

View Diagram

JSON EXAMPLE NOTIFICATION

```
{
 "jsonrpc" : "2.0",
 "method" : "Navigation.OnAudioDataStreaming",
 "params" :
 {
  "available" : true
 }
}
```

OnTBTClientState

Type Notification
Sender HMI
Purpose

Provide SDL with information about changes to the TBT client state.

Notification

PARAMETERS

NAME	TYPE	MANDATORY	ADDITIONAL
state	Common.TBTState	true	

Sequence Diagrams

SEQUENCE DIAGRAM

OnTBTClientState

View Diagram

JSON EXAMPLE NOTIFICATION

```
{
 "jsonrpc" : "2.0",
 "method" : "Navigation.OnTBTClientState",
 "params" :
 {
  "state" : "NEXT_TURN_REQUEST"
 }
}
```

OnVideoDataStreaming

Type Notification
Sender SDL
Purpose

Notify the HMI about a raw video data stream which should be visible to the user while in that application's context.

Notification

PARAMETERS

NAME	TYPE	MANDATORY	ADDITIONAL
available	Boolean	true	

Sequence Diagrams

SEQUENCE DIAGRAM

OnVideoDataStreaming

View Diagram

JSON EXAMPLE NOTIFICATION

```
{
 "jsonrpc" : "2.0",
 "method" : "Navigation.OnVideoDataStreaming",
 "params" :
 {
  "available" : true
 }
}
```

SendLocation

Type Function
Sender SDL
Purpose

> Allow a connected application to send a destination to the embedded navigation system.

Behavior

!!! must
1. Transfer data received from SDL to navigation embedded module. Navigation embedded HU system should update the location data on the embedded navi screen.
2. Send a `SUCCESS` response to SDL in case the destination data has been obtained.
3. Ignore `<deliveryMode>`, `<timeStampparams>`, `<address>` parameters in case HMI does not support them (note: app will not know whether the whole information was processed by HMI)
4. Respond with `WARNINGS` and “info: were not processed”” to SDL in case HMI partially supports `<deliveryMode>`, `<timeStampparams>`, `<address>` (SDL will transfer it to mobile app)
5. Respond with `REJECTED` result code to SDL in case HMI is currently busy

with a higher-priority event or other reasons defined for the mentioned resultCode by HMI.

REQUEST

PARAMETERS

NAME	TYPE	MANDATORY	ADDITIONAL
appID	Integer	true	
longitudeDegrees	Float	false	minvalue: -180 maxvalue: 180
latitudeDegrees	Float	false	minvalue: -90 maxvalue: 90
locationName	String	false	maxlength: 500
locationDescription	String	false	maxlength: 500
addressLines	String	false	array: true minsize: 0 maxsize: 4 maxlength: 500
phoneNumber	String	false	maxlength: 500
locationImage	Common.Image	false	
deliveryMode	Common.DeliveryMode	false	
timeStamp	Common.DateTime	false	
address	Common.OASISAddress	false	

RESPONSE

PARAMETERS

This RPC has no additional parameter requirements

Sequence Diagram

SEQUENCE DIAGRAM

SendLocation Success

View Diagram

SEQUENCE DIAGRAM

SendLocation Fail with warnings

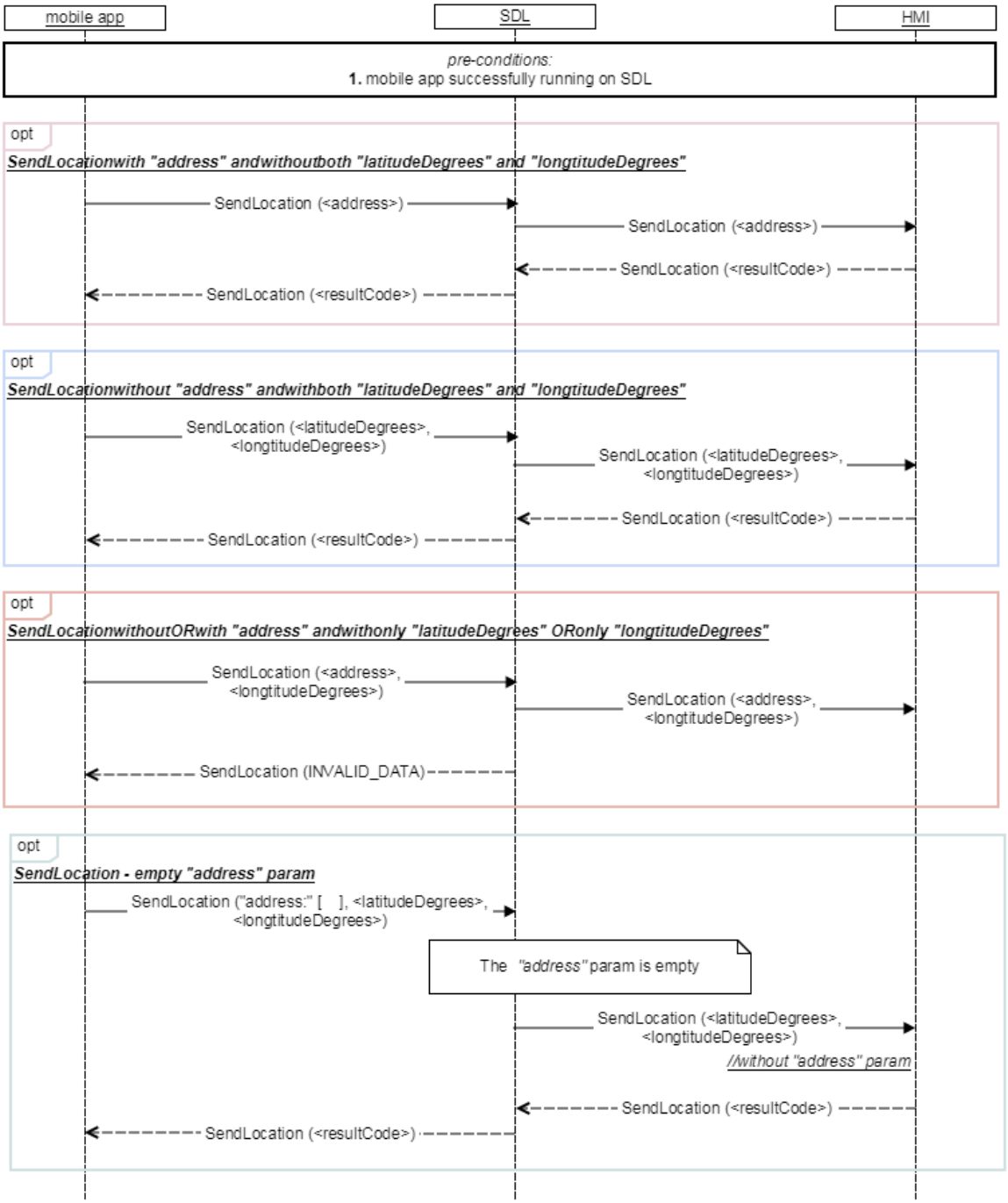
View Diagram

SEQUENCE DIAGRAM

SendLocation Fail Rejected

View Diagram

SendLocation General behavior



|||

Example Request

```
{
 "id" : 138,
 "jsonrpc" : "2.0",
 "method" : "Navigation.SendLocation",
 "params" :
 {
  "longitudeDegrees" : 139.34,
  "latitudeDegrees" : 35.36,
  "locationName" : "Ford Repair",
  "locationImage" :
  {
      "value" : "tmp/SDL/app/Navi/12345.jpg",
      "imageType" : DYNAMIC
  }
 }
}
```

Example Response

```
{
 "id" : 138,
 "jsonrpc" : "2.0",
 "result" :
 {
  "code" : 0,
  "method" : "Navigation.SendLocation"
 }
}
```

Example Error

```
{
 "id" : 138,
 "jsonrpc" : "2.0",
 "error" :
 {
  "code" : 22,
  "message" : " The unknown issue occurred ",
  "data" :
  {
   "method" : "Navigation.SendLocation"
  }
 }
}
```

SetVideoConfig

Type Function
Sender SDL
Purpose Negotiate parameters for a new video stream

`Navigation.SetVideoConfig` represents a request from the app to determine whether a given video streaming configuration is supported by the HMI.

MUST

The HMI must return a `SUCCESS` result code if it is able to handle a stream with the provided parameters.

MUST

The HMI must return a `REJECTED` result code if it is not able to handle a stream with the provided parameters, with any such parameters being included in `rejectedParams`.

REQUEST

PARAMETERS

NAME	TYPE	MANDATORY	ADDITIONAL
config	Common.VideoConfig	true	
appID	Integer	true	

RESPONSE

PARAMETERS

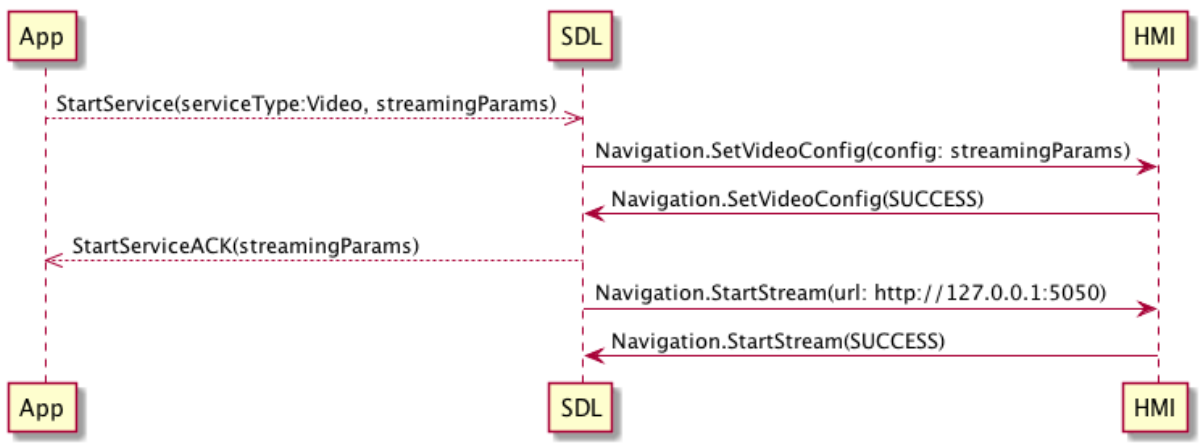
NAME	TYPE	MANDATORY	ADDITIONAL
rejectedParams	String	true	array: true minsize: 1 maxsize: 1000

Sequence Diagrams

SEQUENCE DIAGRAM

SetVideoConfig basic flow

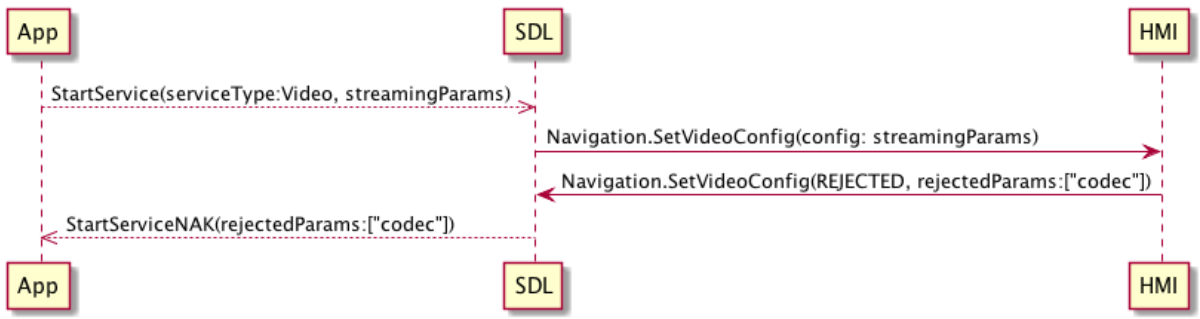
View Diagram



SEQUENCE DIAGRAM

SetVideoConfig error flow

View Diagram



Example Request

```
{
 "id" : 70,
 "jsonrpc" : "2.0",
 "method" : "Navigation.SetVideoConfig",
 "params" :
 {
  "config" :
  {
   "codec": "H264",
   "protocol": "RTP",
   "width": 800,
   "height": 600
  },
  "appID" : 65464
 }
}
```

Example Response

```
{
 "id" : 70,
 "jsonrpc" : "2.0",
 "result" :
 {
  "code" : 0,
  "method" : "Navigation.SetVideoConfig"
 }
}
```

Example Error

```
{
 "id" : 70,
 "jsonrpc" : "2.0",
 "result" :
 {
  "code" : 4,
  "rejectedParams": ["codec", "width"],
  "method" : "Navigation.SetVideoConfig"
 }
}
```

ShowConstantTBT

Type

Function

Sender

SDL

Purpose

Update navigation information of the embedded navigation system.

REQUEST

PARAMETERS

NAME	TYPE	MANDATORY	ADDITIONAL
navigationTexts	Common.TextFieldStruct	true	array: true minsize: 0 maxsize: 5
turnIcon	Common.Image	false	
nextTurnIcon	Common.Image	false	
distanceToManeuver	Float	true	minvalue: 0 maxvalue: 1000000000
distanceToManeuverScale	Float	true	minvalue: 0 maxvalue: 1000000000
maneuverComplete	Boolean	false	
softButtons	Common.SoftButton	false	array: true minsize: 0 maxsize: 3
appID	Integer	true	

RESPONSE

PARAMETERS

This RPC has no additional parameter requirements

Sequence Diagrams

SEQUENCE DIAGRAM

ShowConstantTBT

View Diagram

Example Request

```
{
 "id" : 543,
 "jsonrpc" : "2.0",
 "method" : "Navigation.ShowConstantTBT",
 "params" :
 {
  "navigationTexts" :
  [
   {
    "fieldName" :  navigationText1,
    "fieldText" : "Destination point: Berlin"
   },
   {
    "fieldName" :  ETA,
    "fieldText" : "15:45"
   },
   {
    "fieldName" :  totalDistance,
    "fieldText" : "658"
   }
  ],
  "turnIcon" :
  [
      "value" : "tmp/SDL/app/Navi/icon_3245.jpeg",
      "imageType" : DYNAMIC
  ],
  "distanceToManeuver" : 168,
  "distanceToManeuverScale" : 265,
  "softButtons" :
  [
     "type" :  TEXT,
     "text" : "Close",
     "softButtonID" : 76,
     "systemAction" :  DEFAULT_ACTION
  ],
  "appID" : 26743
 }
}
```

Example Response

```
{
 "id" : 543,
 "jsonrpc" : "2.0",
 "result" :
 {
  "code" : 0,
  "method" : "Navigation.ShowConstantTBT"
 }
}
```

Example Error

```
{
 "id" : 543,
 "jsonrpc" : "2.0",
 "error" :
 {
  "code" : 5,
  "message" : " A command was aborted",
  "data" :
  {
   "method" : "Navigation.ShowConstantTBT"
  }
 }
}
```

StartAudioStream

Type

Function

Sender

SDL

Purpose

Initiate an audio streaming service between SDL and the HMI to stream audio data.

REQUEST

PARAMETERS

NAME	TYPE	MANDATORY	ADDITIONAL
url	String	true	minlength: 21 maxlength: 500
appID	Integer	true	

RESPONSE

PARAMETERS

This RPC has no additional parameter requirements

Sequence Diagrams

SEQUENCE DIAGRAM

StartAudioStream

View Diagram

Example Request

```
{
 "jsonrpc" : "2.0",
 "method" : "Navigation.StartAudioStream",
 "params" :
 {
  "url" : "SDL/application_directory/audio/123.mp3",
  "appID" : 65674
 }
}
```

Example Response


```
{
 "id" : 176,
 "jsonrpc" : "2.0",
 "result" :
 {
  "code" : 0,
  "method" : "Navigation.StartAudioStream"
 }
}
```


Example Error


```
{
 "id" : 176,
 "jsonrpc" : "2.0",
 "error" :
 {
  "code" : 22,
  "message" : "Start stream failed or some other error occurred",
  "data" :
  {
   "method" : "Navigation.StartAudioStream"
  }
 }
}
```

StartStream

Type

Function

Sender

SDL

Purpose

Initiate a video streaming service between SDL and the HMI to stream video data.

REQUEST

PARAMETERS

NAME	TYPE	MANDATORY	ADDITIONAL
url	String	true	minlength: 21 maxlength: 500
appID	Integer	true	

RESPONSE

PARAMETERS

This RPC has no additional parameter requirements

Sequence Diagrams

SEQUENCE DIAGRAM

StartStream

View Diagram

Example Request

```
{
 "jsonrpc" : "2.0",
 "method" : "Navigation.StartStream",
 "params" :
 {
  "url" : "SDL/application_directory/video/123.mp4",
  "appID" : 65674
 }
}
```

Example Response


```
{
 "id" : 176,
 "jsonrpc" : "2.0",
 "result" :
 {
  "code" : 0,
  "method" : "Navigation.StartStream"
 }
}
```


Example Error


```
{
 "id" : 176,
 "jsonrpc" : "2.0",
 "error" :
 {
  "code" : 22,
  "message" : "Start stream failed or some other error occurred",
  "data" :
  {
   "method" : "Navigation. StartStream"
  }
 }
}
```

StopAudioStream

Type

Function

Sender

SDL

Purpose

Stop the audio streaming service between SDL and the HMI and stop streaming audio data.

REQUEST

PARAMETERS

NAME	TYPE	MANDATORY	ADDITIONAL
appID	Integer	true	

RESPONSE

PARAMETERS

This RPC has no additional parameter requirements

Sequence Diagrams

SEQUENCE DIAGRAM

StopAudioStream

View Diagram

Example Request

```
{
 "jsonrpc" : "2.0",
 "method" : "Navigation.StopAudioStream",
 "params" :
 {
  "appID" : 65674
 }
}
```

Example Response


```
{
 "id" : 176,
 "jsonrpc" : "2.0",
 "result" :
 {
  "code" : 0,
  "method" : "Navigation.StopAudioStream"
 }
}
```


Example Error


```
{
 "id" : 176,
 "jsonrpc" : "2.0",
 "error" :
 {
  "code" : 22,
  "message" : "Stop stream failed or some other error occurred",
  "data" :
  {
   "method" : "Navigation.StopAudioStream"
  }
 }
}
```

StopStream

Type Function
Sender SDL
Purpose Close the video streaming service between SDL and the HMI and stop streaming video data.

The app must stop video streaming after receiving onHMIStatus with videoStreamingState=NOT_STREAMBLE.
If the app does not stop streaming after certain amount of time, SDL sends a StopService Control Frame to the app in protocol layer.

PARAMETERS

NAME	TYPE	MANDATORY	ADDITIONAL
appID	Integer	true	

RESPONSE

PARAMETERS

This RPC has no additional parameter requirements

Sequence Diagrams

SEQUENCE DIAGRAM

StopStream

View Diagram

Example Request

```
{
 "jsonrpc" : "2.0",
 "method" : "Navigation.StopStream",
 "params" :
 {
  "appID" : 65674
 }
}
```

Example Response


```
{
 "id" : 176,
 "jsonrpc" : "2.0",
 "result" :
 {
  "code" : 0,
  "method" : "Navigation.StopStream"
 }
}
```


Example Error


```
{
 "id" : 176,
 "jsonrpc" : "2.0",
 "error" :
 {
  "code" : 22,
  "message" : "Stop stream failed or some other error occurred",
  "data" :
  {
   "method" : "Navigation.StopStream"
  }
 }
}
```

UpdateTurnList

Type

Function

Sender

SDL

Purpose

Update embedded navigation's list of directions for the current route.

REQUEST

PARAMETERS

NAME	TYPE	MANDATORY	ADDITIONAL
turnList	Common.Turn	false	array: true minsize: 1 maxsize: 100
softButtons	Common.SoftButton	false	array: true minsize: 0 maxsize: 1
appID	Integer	true	

RESPONSE

PARAMETERS

This RPC has no additional parameter requirements

Sequence Diagrams

SEQUENCE DIAGRAM

UpdateTurnList

View Diagram

Example Request

```
{
 "id" : 176,
 "jsonrpc" : "2.0",
 "method" : "Navigation.UpdateTurnList",
 "params" :
 {
  "turnList" :
  [
   {
    "navigationText" :
    [
     "fieldName" :  navigationText,
     "fieldText" : "Turn Right"
    ],
    "turnIcon" :
    [
      "value" : "tmp/SDL/app/Navi/icon_turn_right.jpeg",
      "imageType" :  DYNAMIC
    ]
   },

   {
    "navigationText" :
    [
     "fieldName" :  navigationText,
     "fieldText" : "Turn Left"
    ],
    "turnIcon" :
    [
      "value" : "tmp/SDL/app/Navi/icon_turn_left.jpeg",
      "imageType" :  DYNAMIC
    ]
   },

   {
    "navigationText" :
    [
     "fieldName" :  navigationText,
     "fieldText" : "Go Forward"
    ],
    "turnIcon" :
    [
      "value" : "tmp/SDL/app/Navi/icon_go_forward.jpeg",
      "imageType" :  DYNAMIC
    ]
   }
  ],

  "softButtons" :
```

```
    [
      "type" :  BOTH,
      "text" : "Return",
      "image" :
      [
        "value" : "tmp/SDL/app/Navi/icon_583.jpg",
        "imageType" : DYNAMIC
      ],
      "isHighlighted" : true,
      "softButtonID" : 118,
      "systemAction" :  DEFAULT_ACTION
    ]
  }
}
```

Example Response

```
{
  "id" : 176,
  "jsonrpc" : "2.0",
  "result" :
  {
    "code" : 0,
    "method" : "Navigation.UpdateTurnList"
  }
}
```

Example Error

```
{
  "id" : 176,
  "jsonrpc" : "2.0",
  "error" :
  {
    "code" : 4,
    "message" : " A command was rejected because a higher priority
command is requested",
    "data" :
    {
      "method" : "Navigation.UpdateTurnList"
    }
  }
}
```

ActivateApp

Type Function
Sender HMI
Purpose Inform SDL that the user has activated an application.

REQUEST

MUST

1. Send `SDL.ActivateApp`.
2. Send a request to SDL to get messages for specified permissions (via `GetUserFriendlyMessage`) and notify user that provided permissions of application were decreased in case HMI gets "isAppPermissionRevoked:true" respond from SDL PoliciesManager.
3. Send `GetListOfPermissions` request to SDL in order to obtain list of message codes for functional groups needed by application for user to consent when PoliciesManager responds with "isPermissionsConsentNeeded: true" .
4. Display Dialog and on result of user selection to send `OnAllowSDLFuncionality` specifying device from `ActivateApp` response, source of choice (UI/VR) and allowed set to true/false (if user ignores question, this is automatically set to false) when HMI receives `SDL.ActivateApp` (isSDLAllowed: false).

NOTE

SDL PoliciesManager must:

- respond with "isPermissionsConsentNeeded: true", otherwise "isPermissionsConsentNeeded: false" should be returned to HMI, in case SDL got *SDL.ActivateApp* from HMI and LocalPT contains the permission that require User`s consent.
- respond with "isSDLAllowed: true" in the response to HMI, on getting *SDL.ActivateApp* from HMI and the device the app is running on is consented by the User.
- respond with the following parameters in the response to HMI:
  - "isSDLAllowed: false"
  - *device* param containing the device`s name and ID previously sent by SDL via *UpdateDeviceList*.

PARAMETERS

NAME	TYPE	MANDATORY	DESCRIPTION
appID	Integer	true	ID of the application that the User has chosen.

RESPONSE

PARAMETERS

NAME	TYPE	MANDATORY	DESCRIPTION
isSDLAllowed	Boolean	true	SDL returns:‘true’, in case the User has allowed using the device for PolicyTable Exchange. ‘false’, in case the User has not yet been asked for or in case the User has disallowed using the device for PolicyTable Exchange.
device	Common.DeviceInfo	false	
isPermissionsConsentNeeded	Boolean	true	
isAppPermissionsRevoked	Boolean	true	
appRevokedPermissions	Common.PermissionItem	false	array: true minsize: 1 maxsize: 100
isAppRevoked	Boolean	true	
priority	Common.AppPriority	false	

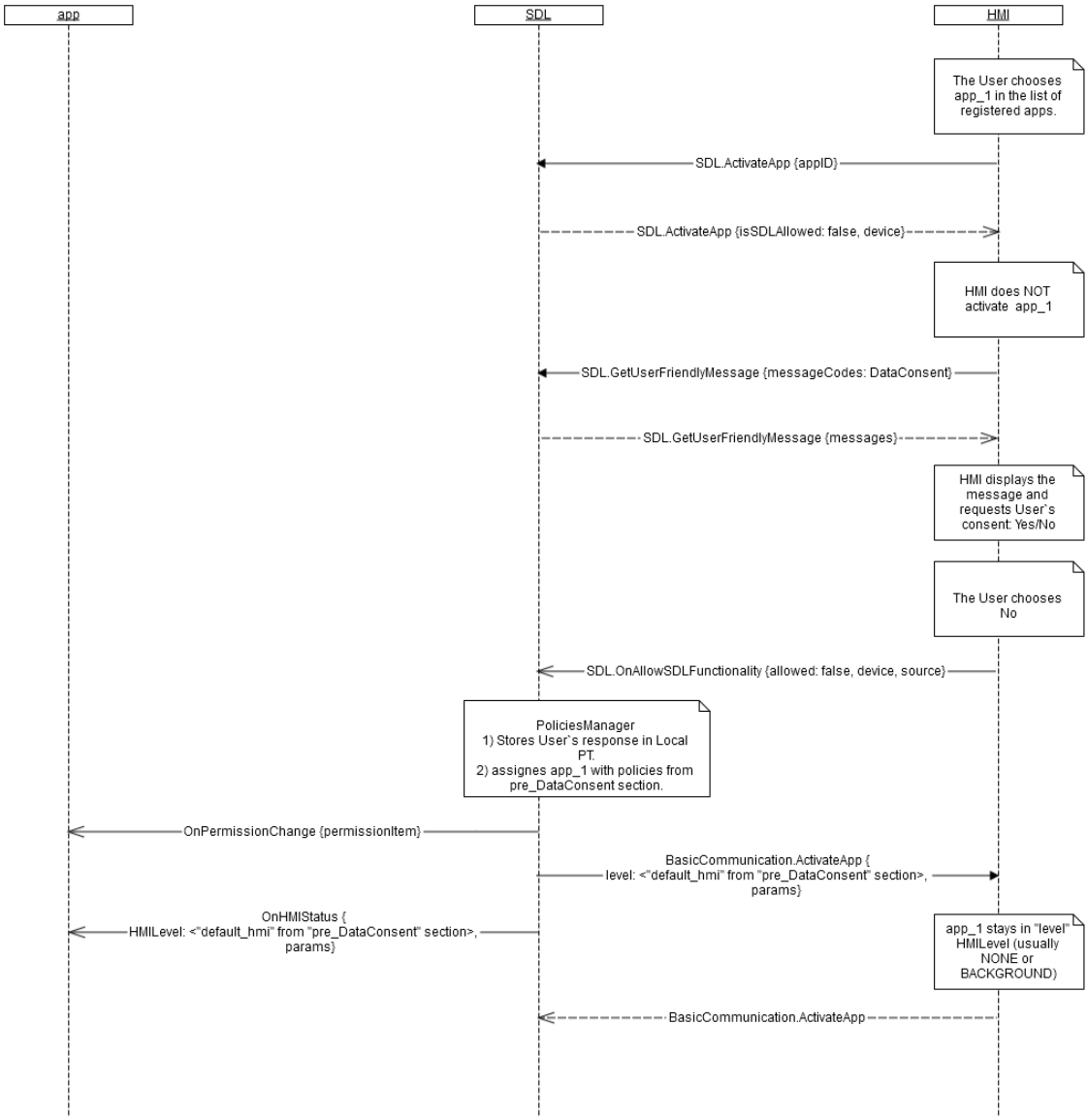
Sequence Diagrams

Preconditions:
a. Device is connected to the HU.
b. App_1 is running on this device and is registered with SDL.
c. App_1 presents in the list of registered apps on HMI.

SEQUENCE DIAGRAM

The User does NOT consent the device.

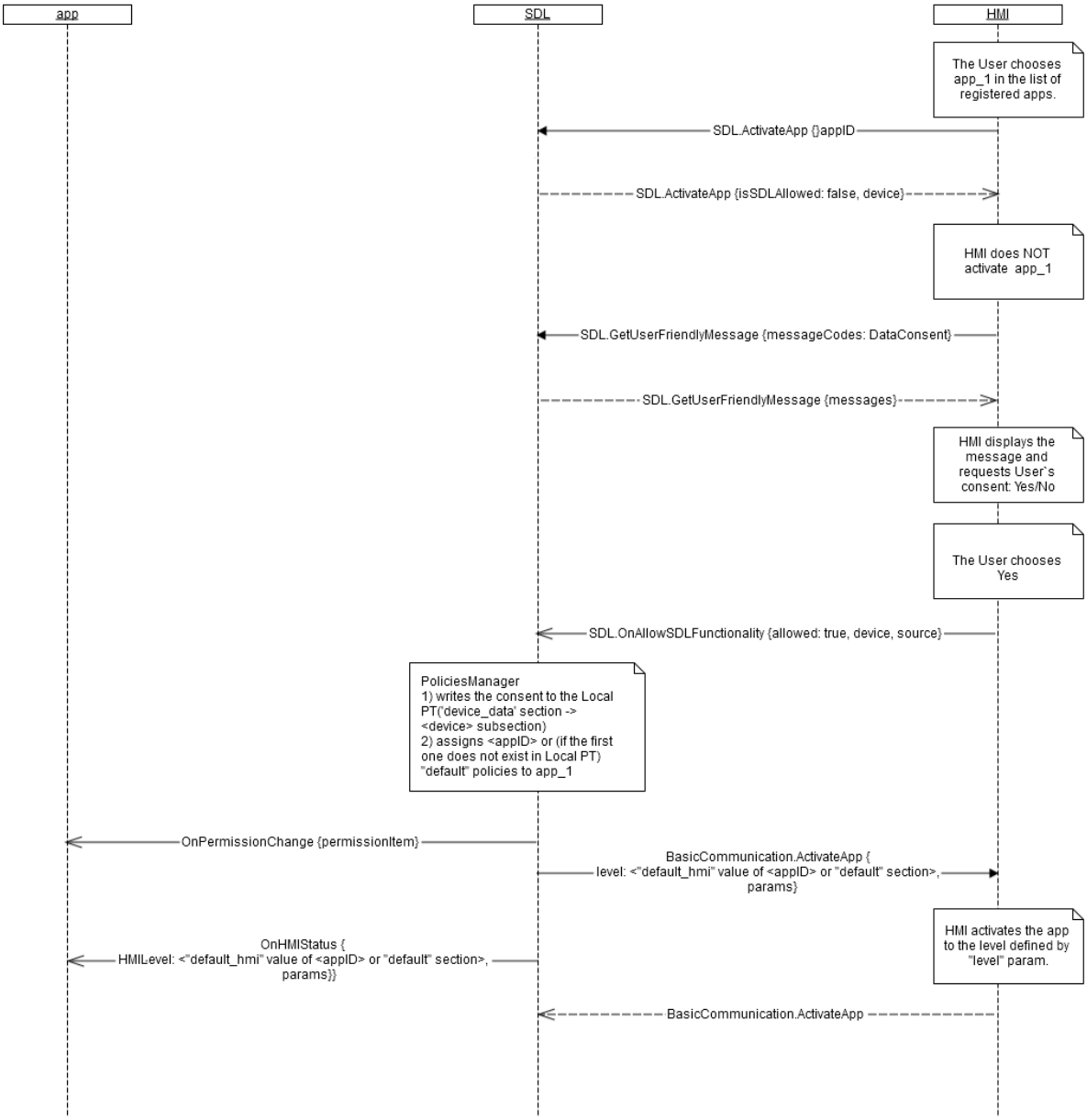
View Diagram



SEQUENCE DIAGRAM

The User consents the device.

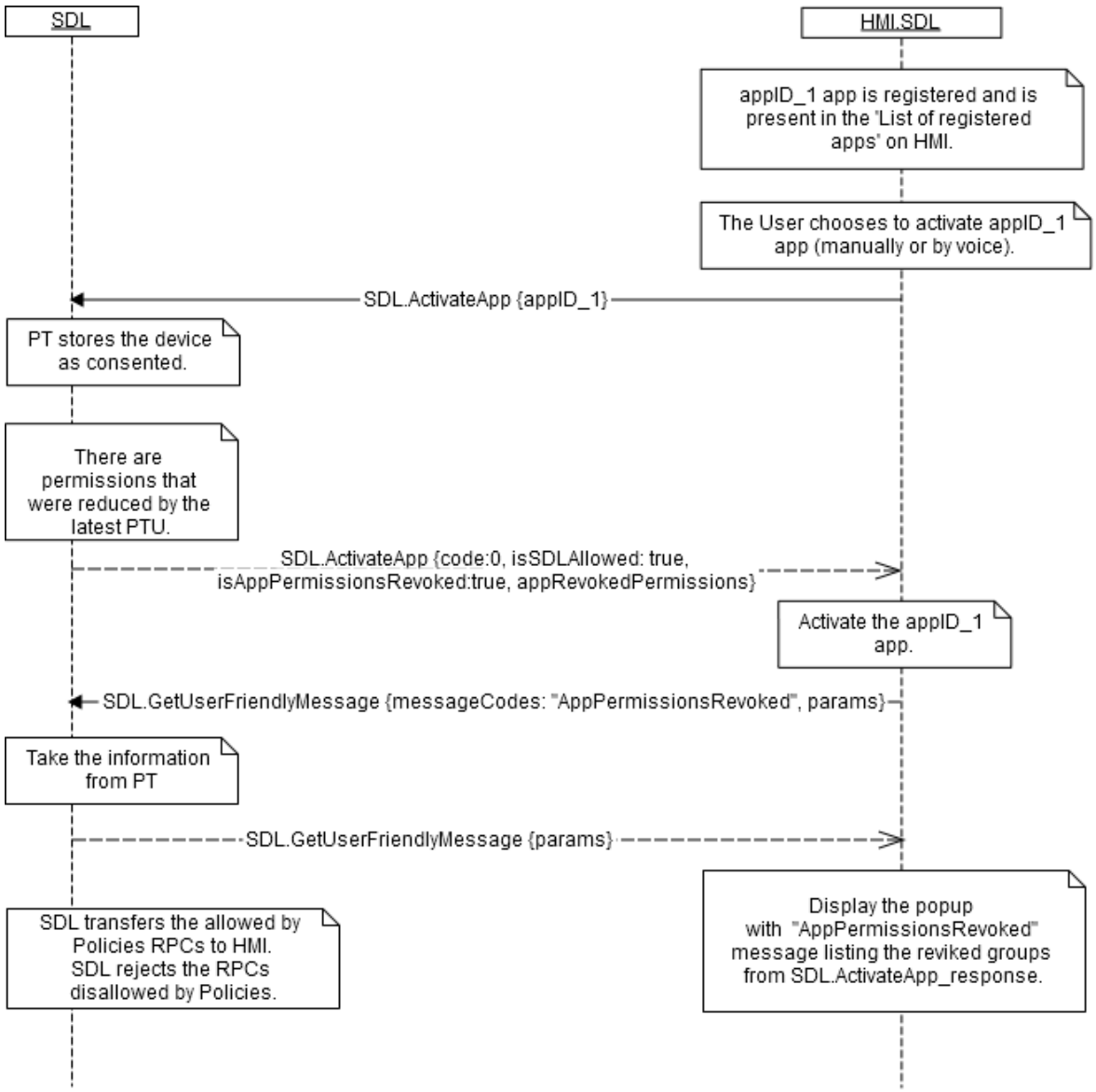
View Diagram



SEQUENCE DIAGRAM

ActivateApp for application registered with reduced permissions after Policy Table Update

View Diagram

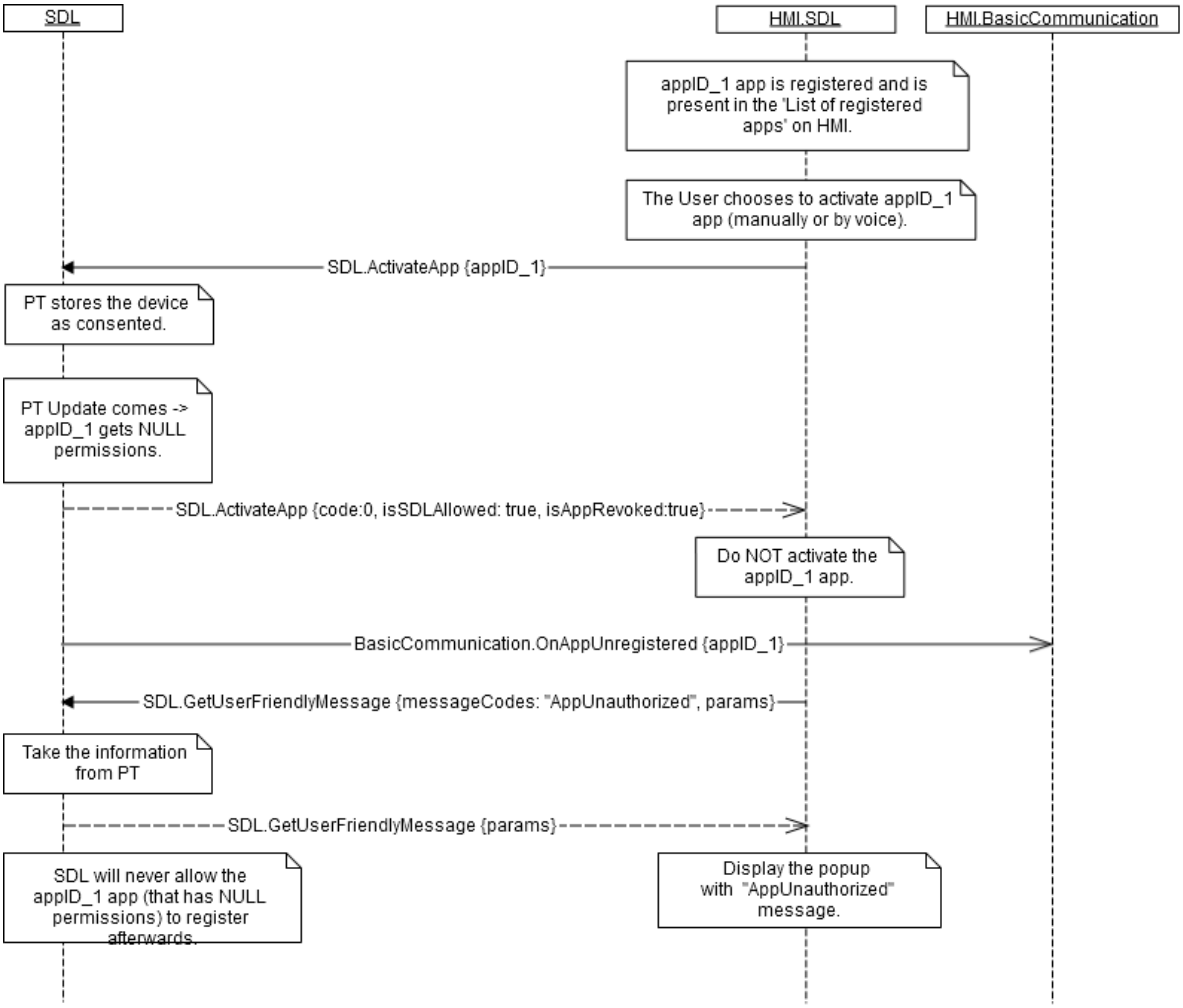


SEQUENCE DIAGRAM

ActivateApp for application registered with revoked permissions after Policy Table Update

---

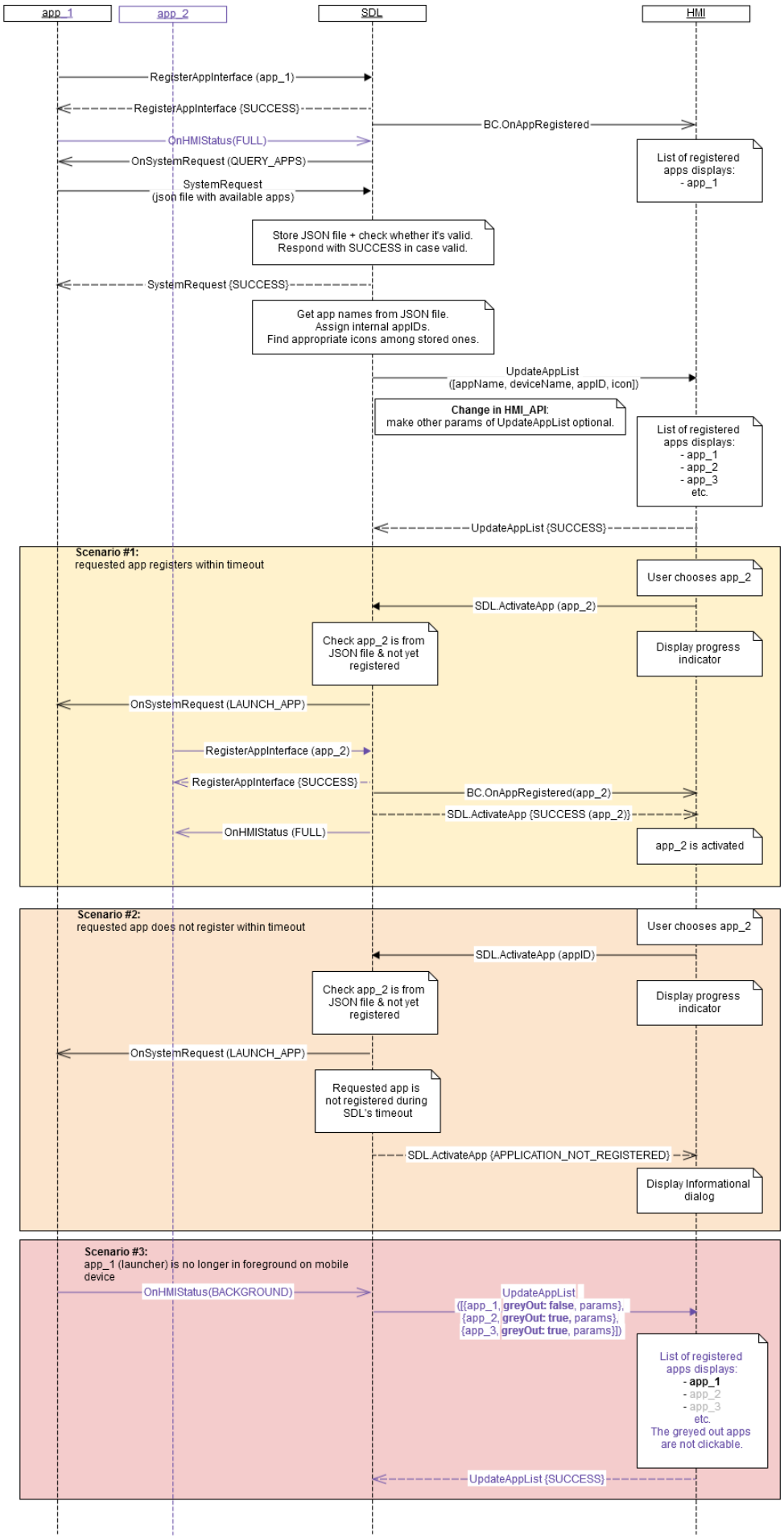
View Diagram



SEQUENCE DIAGRAM

ActivateApp using App Launching

View Diagram


app_1
app_2
SDL
HMI
RegisterAppInterface (app_1)
RegisterAppInterface {SUCCESS}
BC.OnAppRegistered
OnHMIStatus(FULL)
List of registered apps displays: - app_1
OnSystemRequest (QUERY_APPS)
SystemRequest (json file with available apps)
Store JSON file + check whether it's valid. Respond with SUCCESS in case valid.
SystemRequest {SUCCESS}
Get app names from JSON file. Assign internal appIDs. Find appropriate icons among stored ones.
UpdateAppList ([appName, deviceName, appID, icon])
Change in HMI_API: make other params of UpdateAppList optional.
List of registered apps displays: - app_1 - app_2 - app_3 etc.
UpdateAppList {SUCCESS}
Scenario #1: requested app registers within timeout
User chooses app_2
SDL.ActivateApp (app_2)
Check app_2 is from JSON file & not yet registered
Display progress indicator
OnSystemRequest (LAUNCH_APP)
RegisterAppInterface (app_2)
RegisterAppInterface {SUCCESS}
BC.OnAppRegistered(app_2)
SDL.ActivateApp {SUCCESS (app_2)}
OnHMIStatus (FULL)
app_2 is activated
Scenario #2: requested app does not register within timeout
User chooses app_2
SDL.ActivateApp (appID)
Check app_2 is from JSON file & not yet registered
Display progress indicator
OnSystemRequest (LAUNCH_APP)
Requested app is not registered during SDL's timeout
SDL.ActivateApp {APPLICATION_NOT_REGISTERED}
Display Informational dialog
Scenario #3: app_1 (launcher) is no longer in foreground on mobile device
OnHMIStatus(BACKGROUND)
UpdateAppList ([{app_1, greyOut: false, params}, {app_2, greyOut: true, params}, {app_3, greyOut: true, params}])
List of registered apps displays: - app_1 - app_2 - app_3 etc. The greyed out apps are not clickable.
UpdateAppList {SUCCESS}

Example Request

```
{
 "id" : 27,
 "jsonrpc" : "2.0",
 "method" : "SDL.ActivateApp"
 "params" :
 {
  "appID" : 12345
 }
}
```

Example Response

```
{
 "id" : 27,
 "jsonrpc" : "2.0",
 "result" :
 {
  "isSDLAllowed" : true,
  "isPermissionsConsentNeeded" : false,
  "isAppPermissionsRevoked" : false,
  "isAppRevoked" : false,
  "code" : 0,
  "method" : "SDL.ActivateApp"
 }
}
```

Example Error

```
{
 "id" : 27,
 "jsonrpc" : "2.0",
 "error" :
 {
  "code" : 22,
  "message" : "The unknown error has occurred",
  "data" :
  {
   "method" : "SDL.ActivateApp"
  }
 }
}
```

AddStatisticsInfo

Notification

PARAMETERS

NAME	TYPE	MANDATORY	ADDITIONAL
statisticType	Common.StatisticsType	true	

Example Request

Example Response

Example Error

GetListOfPermissions

Type Function	
Sender	

HMI
Purpose Get the list of permissions for the specified application or for all applications.

REQUEST

!!! MUST

1) Initiate sending GetListOfPermissions with “appID” parameter in the following cases:
1.1) After receiving SDL.ActivateApp{isPermissionsConsentNeeded: true} from SDL.
Note: appID is known from the corresponding SDL.ActivateApp request.
1.2) After receiving SDL.OnAppPermissionChanged{appID, appPermissionsConsentNeeded: true} from SDL.
1.3) After the User presses the button to change the permissions for the application.
1.4) After ignition on to retrieve the "External UCS Status" setting stored on SDL.

2) Initiate sending GetListOfPermissions without “appID” parameter in the following cases:
2.1) After the User presses the button to change the permissions of all currently registered applications.

Impotant:
When GetListOfPermissions is requested without “appID” parameter, the response of PermissionItem array contains the PermissionGroups (“name” parameters) for all the applications as a whole and “allowed” parameter is common for each group (e.g. if at least one of the applications has disallowed parameter for some group, “allowed” value returned for this group in the context of all application must be false). (*For more details see the diagram*

GetListOfPermissions without appID).
3) Store the pair of values id<->name of PermissionItem structure which were obtained via GetListOfPermissions response. These pairs will be used for future notifications via OnAppPermissionConsent in case user's choice for the application to allow some functionality has been updated (*For more details see* OnAppPermissionConsent (id<->name dependency).
4) Display the received "External UCS Status" in the appropriate UI screen (*For more details see the picture ExternalConsentStatus*).

!!! NOTE
a) The information about the application (name, ID, etc.) is provided by SDL via either BasicCommunication.UpdateAppList or BasicCommunication.OnAppRegistered RPCs.
b) If HMI has never sent externalConsentStatus before (via OnAppPermissionConsent), SDL will respond with empty array.
c) User can go to settings on HMI and enable or disable "External UCS" for an "entity". HMI will notify SDL accordingly (*see figure ExternalConsentStatus*).
d) The resulting allowance of functional grouping is informed by parameter "allowedFunctions" to HMI in accordinance with result of ExternalConsentStatus.

PARAMETERS

NAME	TYPE	MANDATORY	ADDITIONAL
appID	Integer	false	

RESPONSE

PARAMETERS

NAME	TYPE	MANDATORY	ADDITIONAL
allowedFunctions	Common.PermissionItem	true	array: true minsize: 0 maxsize: 100
externalConsentStatus	Common.ExternalConsentStatus	true	array: true minsize: 0 maxsize: 100

Sequence Diagrams

SEQUENCE DIAGRAM

GetListOfPermissions

View Diagram

SEQUENCE DIAGRAM

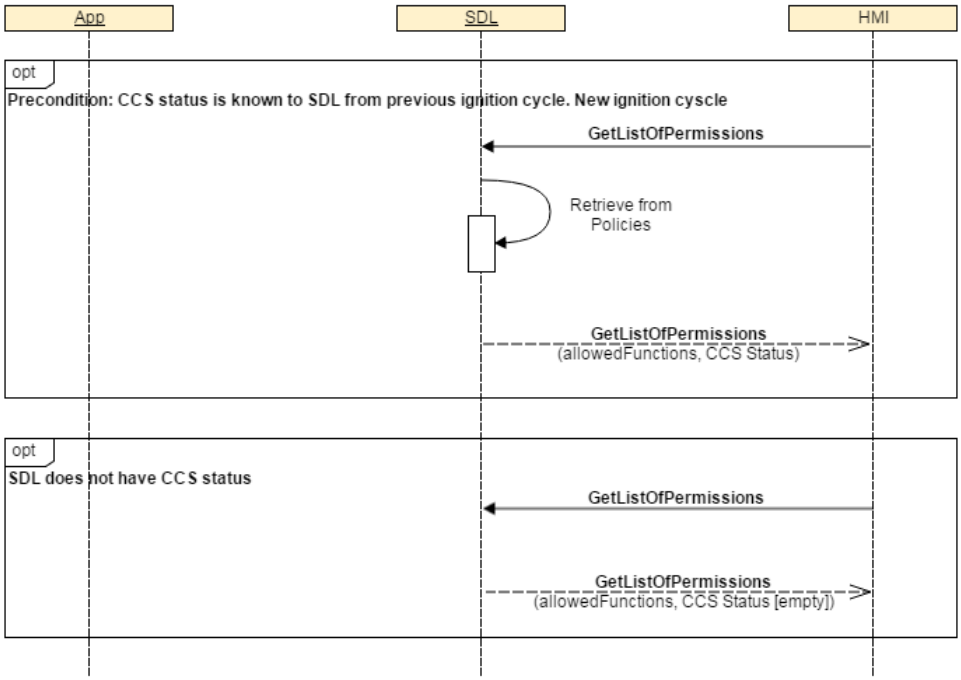
GetListOfPermissions without AppId

View Diagram

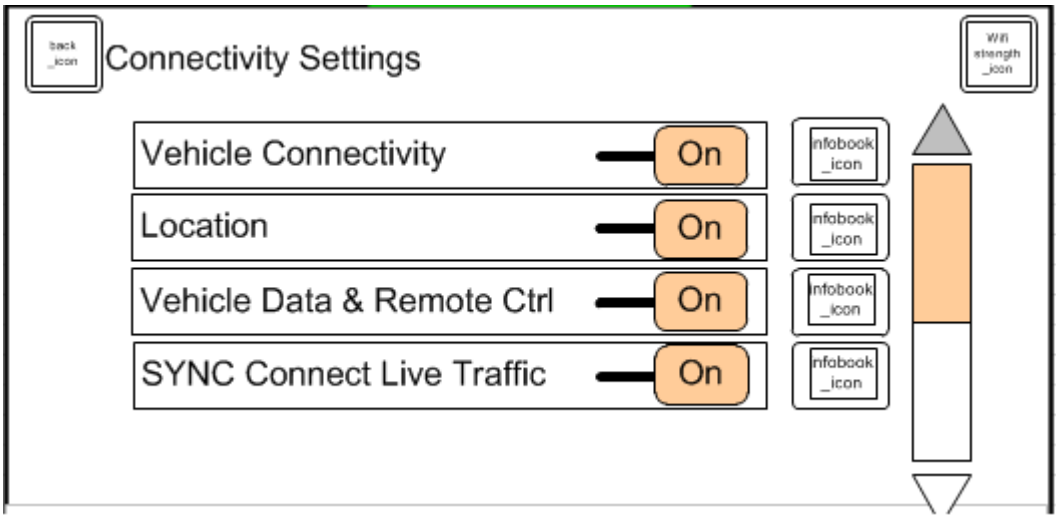
SEQUENCE DIAGRAM

GetListOfPermissions provide External UCS Status to HMI

View Diagram



Possible Layout - ExternalConsentStatus



Example Request

```
{
 "id" : 143,
 "jsonrpc" : "2.0",
 "method" : "SDL.GetListOfPermissions",
 "params" :
 {
  "appID" : 65596
 }
}
```

Example Response


```
{
 "id" : 143,
 "jsonrpc" : "2.0",
 "result" :
 {
  "allowedFunctions" :

[
    {
   "name" : "Location-1",
    "id":1234,
    "allowed":true
 },

   {
   "name" : "Notifications",
    "id":76876,
    "allowed":false
 },

  ]
    "externalConsentStatus":
[
    {
    “entityType” : “0”,
    “entityID”   : “126”,
    “status” : “ON”
    }
]


   "code" : 0,
   "method" : "SDL.GetListOfPermissions"
 }
}
```

Example Error

```
{
  "id" : 143,
  "jsonrpc" : "2.0",
  "error" :
  {
    "code" : 15,
    "message" : " A command cannot be executed because there is NO specified with appID application registered ",
    "data" :
    {
      "method" : "SDL.GetListOfPermissions"
    }
  }
}
```

GetStatusUpdate

Type Function
Sender HMI
Purpose Get information about the current status of the policy update process.

REQUEST

In case HMI needs to find out current status of PTU and sends *GetStatusUpdate_request* to SDL, it must respond with the current update status code to HMI.
The request *GetStatusUpdate* duplicates the functionality of the notification *OnStatusUpdate*. In case the policy update status is being changed.(e.g. an update is finished successfully or retry strategy failed), SDL must send notification *OnStatusUpdate* to HMI with the corresponding UpdateStatus code, whereas *GetStatusUpdate* allowes to request the status of policy table at any time, not on update only.

MUST

Send a request to SDL if it needs to get a current policy update status according to its workflows.

PARAMETERS

This RPC has no additional parameter requirements.

RESPONSE

PARAMETERS

NAME	TYPE	MANDATORY
status	Common.UpdateResult	true

Sequence Diagrams

SEQUENCE DIAGRAM

GetStatusUpdate

View Diagram

Example Request

```
{
 "id" : 176,
 "jsonrpc" : "2.0",
 "method" : "SDL.GetStatusUpdate",
}
```

Example Response

```
{
 "id" : 176,
 "jsonrpc" : "2.0",
 "result" :
 {
  "status" : "UPDATE_NEEDED"
  "code" : 0,
  "method" : "SDL.GetStatusUpdate"
 }
}
```

Example Error

```
{
 "id" : 176,
 "jsonrpc" : "2.0",
 "error" :
 {
  "code" :22,
  "message" : "Some error occured",
  "data" :
  {
   "method" : "SDL.GetStatusUpdate"
  }
 }
}
```

GetURLS

Type Function
Sender HMI
Purpose Retrieve the destination URL(s) from Local Policy Table

REQUEST

MUST

For Proprietary PTU flow:
1. Send `SDL.GetURLS` after SDL-initiated BC.PolicyUpdate with service type `7`
2. Use the received URL(s) in the next BC.OnSystemRequest.

MAY

Request the URL(s) for any known service type that exists in Local PT at any time if required per internal flows.

NOTE

`SDL.GetURLS` dependencies:

- SDL sends `BC.PolicyUpdate` *only in case* it's built with "-DEXTENDED_POLICY: PROPRIETARY" flag or without flag.
- URLs storage in Policy Table: `"endpoints"` section.
- SDL chooses the applications for taking part in PTU among registered ones and provides `appID` + `url` pairs in response
- In case no applications are currently registered, SDL will return `url` only.

PARAMETERS

NAME	TYPE	MANDATORY	ADDITIONAL
service	Integer	true	minvalue: 0 maxvalue: 100

RESPONSE

PARAMETERS

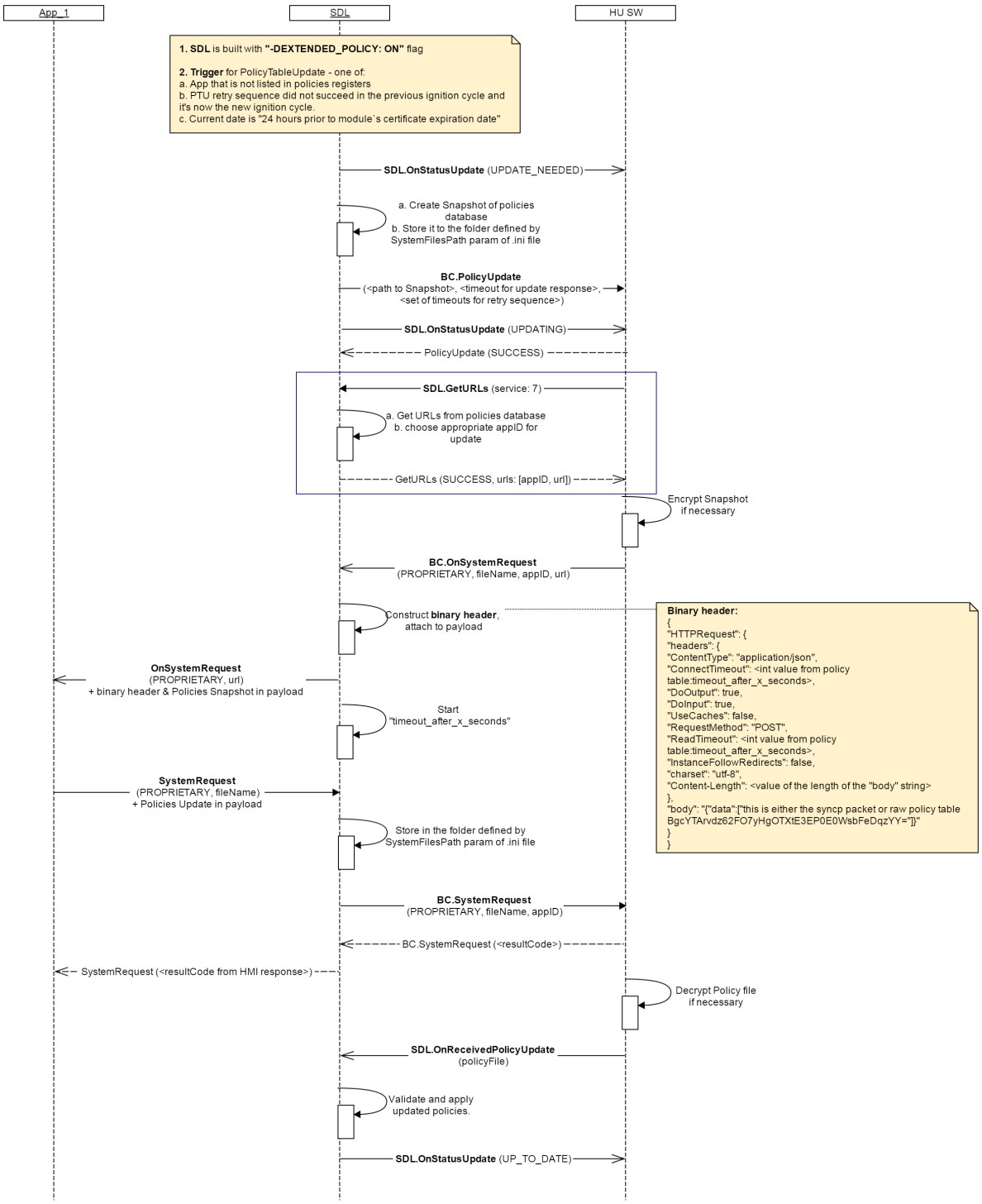
NAME	TYPE	MANDATORY	ADDITIONAL
urls	Common.ServiceInfo	false	array: true minsize: 1 maxsize: 100

Sequence Diagrams

SEQUENCE DIAGRAM

BC.PolicyUpdate in "Proprietary" Policy Table Update Flow

View Diagram


App_1
SDL
HU SW
1. SDL is built with "-DEXTENDED_POLICY: ON" flag
2. Trigger for PolicyTableUpdate - one of:
a. App that is not listed in policies registers
b. PTU retry sequence did not succeed in the previous ignition cycle and it's now the new ignition cycle.
c. Current date is "24 hours prior to module`s certificate expiration date"
SDL.OnStatusUpdate (UPDATE_NEEDED)
a. Create Snapshot of policies database
b. Store it to the folder defined by SystemFilesPath param of .ini file
BC.PolicyUpdate
(<path to Snapshot>, <timeout for update response>, <set of timeouts for retry sequence>)
SDL.OnStatusUpdate (UPDATING)
PolicyUpdate (SUCCESS)
SDL.GetURLs (service: 7)
a. Get URLs from policies database
b. choose appropriate appID for update
GetURLs (SUCCESS, urls: [appID, url])
Encrypt Snapshot if necessary
BC.OnSystemRequest
(PROPRIETARY, fileName, appID, url)
Construct binary header, attach to payload
Binary header:
{
"HTTPRequest": {
"headers": {
"ContentType": "application/json",
"ConnectTimeout": <int value from policy table:timeout_after_x_seconds>,
"DoOutput": true,
"DoInput": true,
"UseCaches": false,
"RequestMethod": "POST",
"ReadTimeout": <int value from policy table:timeout_after_x_seconds>,
"InstanceFollowRedirects": false,
"charset": "utf-8",
"Content-Length": <value of the length of the "body" string>
},
"body": "{"data":["this is either the syncp packet or raw policy table
BgcYTArvdz62FO7yHgOTXtE3EP0E0WsbFeDqzYY="]}"
}
}
OnSystemRequest
(PROPRIETARY, url)
+ binary header & Policies Snapshot in payload
Start "timeout_after_x_seconds"
SystemRequest
(PROPRIETARY, fileName)
+ Policies Update in payload
Store in the folder defined by SystemFilesPath param of .ini file
BC.SystemRequest
(PROPRIETARY, fileName, appID)
BC.SystemRequest (<resultCode>)
SystemRequest (<resultCode from HMI response>)
Decrypt Policy file if necessary
SDL.OnReceivedPolicyUpdate
(policyFile)
Validate and apply updated policies.
SDL.OnStatusUpdate (UP_TO_DATE)

Example Request

```
{
   "id" : 176,
   "jsonrpc" : "2.0",
   "method" : "SDL.GetURLS",
   "params" :
   {
      "service" : 7
   }
}
```

Example Response

```
{
   "id" : 176,
   "jsonrpc" : "2.0",
   "result" :
   {
      "urls" :
      [
       {
         "url" : "https://policies.smartdevicelink.com/api/1/policies",
         "appID" : 65782
       }
      ],
      "code" : 0,
      "method" : "SDL.GetURLS"
   }
}
```

Example Error

```
{
   "id" : 176,
   "jsonrpc" : "2.0",
   "error" :
   {
      "code" : 11,
      "message" : "Invalid data",
      "data" :
      {
         "method" : "SDL.GetURLS"
      }
   }
}
```

GetUserFriendlyMessage

Type Function
Sender HMI
Purpose Get user friendly messages from Policy Table

REQUEST

MUST

- Request message text to notify user via UI or/and TTS message about some event is happening. Message text may also be requested for some specific dialogs on HMI which require user’s answers.
- Notify user according to HMI flow via UI pop-ups or/and TTS messages, the text for them obtained in GetUserFriendlyMessage response in correspondence to messageCodes requested.

NOTE

PoliciesManager must follow to the next steps:
1. In case HMI respond about supported `<language_1>` via *UI.GetLanguage_response* to SDL
and HMI sends *SDL.GetUserFriendlyMessage* request `<language_2>` parameter to SDL and both `<language_1>` and `<language_2>` does NOT exist at "consumer_friendly_messages" section at LocalPT:
- retrieve message of en-us sub-section of "consumer_friendly_message" of LocalPT;
- provide this message via *SDL.GetUserFriendlyMessage_response* to HMI.
2. In case HMI respond about supported `<language_1>` via *UI.GetLanguage_response* to SDL and HMI sends *SDL.GetUserFriendlyMessage* request `<language_2>` parameter to SDL and requested `<language_2>` does NOT exist at "consumer_friendly_messages" section at LocalPT
and `<language_1>` exists at "consumer_friendly_messages" section at LocalPT:
- retrieve message of `<language_1>` from "consumer_friendly_message" of LocalPT;
- provide this message via *SDL.GetUserFriendlyMessage_response* to HMI.
3. In case HMI sends *SDL.GetUserFriendlyMessage request* with `<language_1>` parameter and this `<language_1>` exists at "consumer_friendly_message" section of LocalPT:
- retrieve message related to requested `<language_1>` from "consumer_friendly_message" section of LocalPT;
- provide this message via GetUserFriendlyMessage response to HMI.

PARAMETERS

NAME	TYPE	MANDATORY	ADDITIONAL
messageCodes	String	true	array: true minsize: 1 maxsize: 100 maxlength: 500
language	Common.Language	false	

MessageCodes

Message codes specify appropriate user messages which notify the user about some events/conditions on HMI. Messages and message codes are coming from Policy Table and must be used on HMI in different information pop-ups according to HMI use-cases scenarios.

MESSAGECODE	MESSAGETEXT
AppPermissions	appName is requesting the use of the following vehicle information and permissions functionalGroupLabels. Grant requested permission(s)
AppPermissionsHelp	appName is requesting the following vehicle information and permissions: functionalGroupLabels. You can change these permissions and hear detailed descriptions in the mobile apps settings menu. Please press 'yes' to grant permissions or 'no' to deny.
AppPermissionsRevoked	App authorizations have changed. appName can no longer access functionalGroupLabels. Please, make sure you have the most recent app version installed on your mobile device.
AppUnauthorized	This version of appName is not authorized and will not work with Applink. Please update to the latest version of appName.
AppUnsupported	Your version of appName is not supported by SYNC.
DataConsent	Would you like to enable Mobile Apps on SYNC? See your Owner Guide for more information.
DataConsentHelp	Updates are about the size of an email, and the occurrence of updates depends on your vehicle usage and when a new app is found on your device. See your Owner Guide for more information.
DisableApps	Disabling automatic updates will also disable sync mobile apps. You will not be able to use any mobile apps with SYNC. Please press yes to confirm or no to cancel.
DrivingCharacteristics	An app can access the following driving characteristics: Fuel Consumption, MyKey, Seat Belt Status.

MESSAGECODE	MESSAGETEXT
Location	An app can access vehicle GPS and speed.
Notifications	An app can send notifications when running in the background.
SettingDisableUpdates	Disable Updates
SettingEnableUpdates	Enable Apps
SettingUpdateAuto	Select `Update now` to receive app authorization information for your SYNC-enabled mobile apps. This may enable additional functionality depending on the app and your settings. If your phone has a working data connection, an update should complete in less than 1 minute.
StatusNeeded	Update Needed
StatusPending	Updating...
StatusUpToDate	Up-To-Date
VehicleInfo	An app can access the following vehicle information: Fuel Level, Fuel Economy, Engine RPMs, Odometer, VIN, External Temperature, Gear Position, Tire Pressure.

NOTE

“MessageText” column specifies the messages just for an information purposes. The MessageText received from SDL may differ from listed in the table.

RESPONSE

PARAMETERS

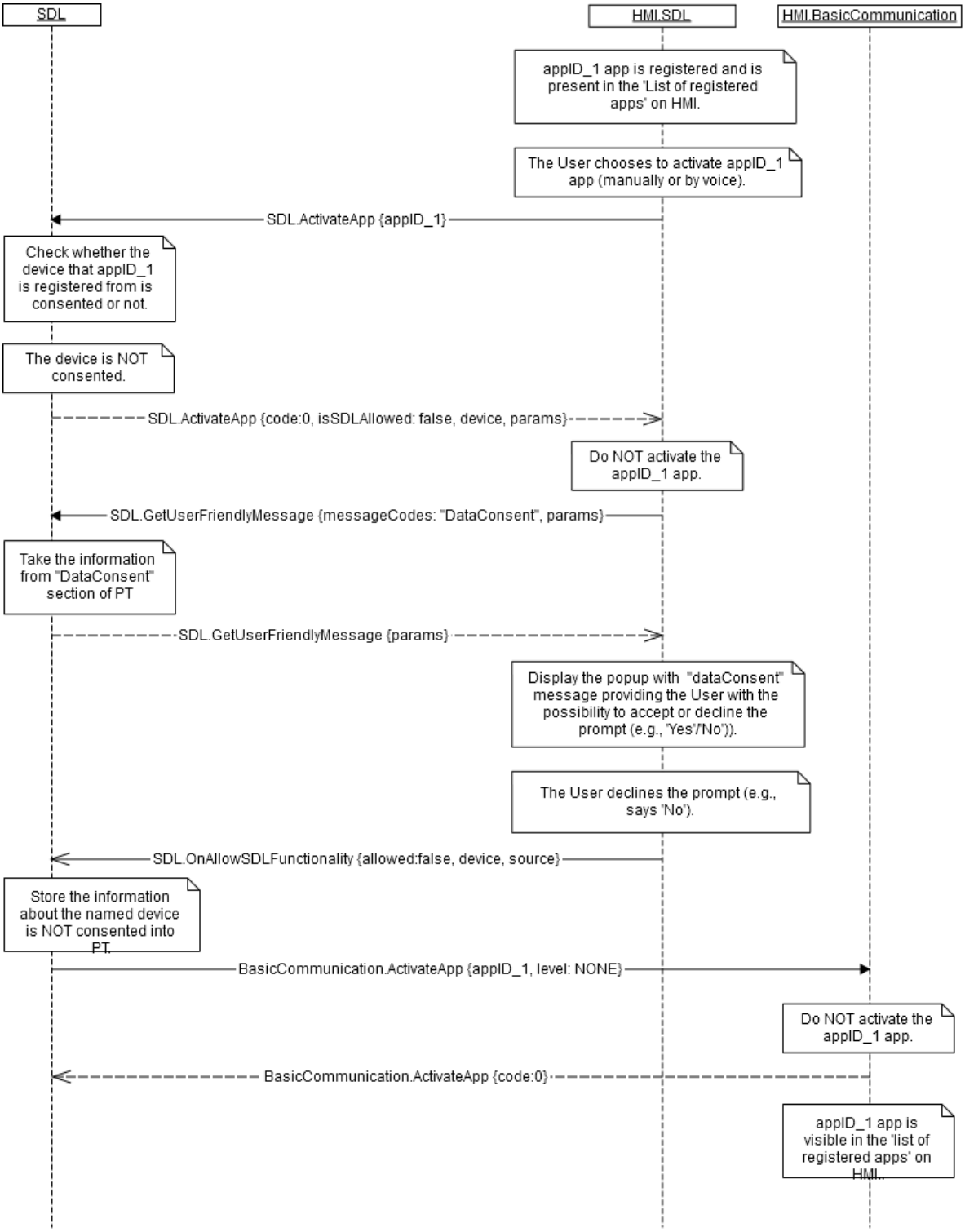
NAME	TYPE	MANDATORY	ADDITIONAL
messages	Common.UserFriendlyMessage	false	array: true minsize: 1 maxsize: 100

Sequence Diagrams

SEQUENCE DIAGRAM

GetUserFriendlyMessage for device consent

View Diagram


SDL
HMI.SDL
HMI.BasicCommunication
appID_1 app is registered and is present in the 'List of registered apps' on HMI.
The User chooses to activate appID_1 app (manually or by voice).
SDL.ActivateApp {appID_1}
Check whether the device that appID_1 is registered from is consented or not.
The device is NOT consented.
SDL.ActivateApp {code:0, isSDLAllowed: false, device, params}
Do NOT activate the appID_1 app.
SDL.GetUserFriendlyMessage {messageCodes: "DataConsent", params}
Take the information from "DataConsent" section of PT
SDL.GetUserFriendlyMessage {params}
Display the popup with "dataConsent" message providing the User with the possibility to accept or decline the prompt (e.g., 'Yes'/'No')).
The User declines the prompt (e.g., says 'No').
SDL.OnAllowSDLFunctionality {allowed:false, device, source}
Store the information about the named device is NOT consented into PT.
BasicCommunication.ActivateApp {appID_1, level: NONE}
Do NOT activate the appID_1 app.
BasicCommunication.ActivateApp {code:0}
appID_1 app is visible in the 'list of registered apps' on HMI.

Example Request

```
{
 "id" : 176,
 "jsonrpc" : "2.0",
 "method" : "SDL.GetUserFriendlyMessage",
 "params" :
 {
  "messageCodes": "AppPermissions",
   "language" : "EN-GB"
 }

}
```

Example Response

```
{
 "id" : 176,
 "jsonrpc" : "2.0",
 "result" :
 {
  "messages": {
          "messageCode": "AppPermissions",
          "ttsString": "%appName% is requesting the use of the
following ….",
          line1: "Grant Requested",
          line2: "Permission(s)?"
         },
  "code" : 0,
  "method" : "SDL.GetUserFriendlyMessage"
 }
}
```

Example Error

```
{
 "id" : 176,
 "jsonrpc" : "2.0",
 "error" :
 {
  "code" :22,
  "message" : "Some error occurred",
  "data" :
  {
   "SDL.GetUserFriendlyMessage"
  }
 }
}
```

OnAllowSDLFunctionality

Type Notification
Sender HMI
Purpose Inform about allowing SDL functionality

Notification

Notifies about user/HMI allowing SDL functionality or disallowing access to all mobile apps on the specified device.

MUST

Send `OnAllowSDLFuntionality` notification to SDL as a result of user’s choice about device data consent via pop-up dialog or user’s settings change in Settings Menu (if applicable)

NOTE

1. SDL ignores all invalid notifications which come from HMI (invalid JSON, invalid data types/bounds etc).
2. In case SDL PoliciesManager receives *SDL.OnAllowSDLFunctionality* with *allowed=false* and without *device* param from HMI, PoliciesManager must record all of currently registered devices as NOT consented in Local PT.
3. In case PoliciesManager receives *SDL.OnAllowSDLFunctionality* with *allowed=true* and without *device* param from HMI, PoliciesManager must:
4. record all of currently registered devices as consented in Local PT;
5. send *BC.ActivateApp*(params, level: ”default_hmi”-value-from-assigned-policies) to HMI (even if this “default_hmi” value is NONE or BACKGROUND).
6. notify mobile application via *OnHMIStatus*(params, level: ”default_hmi”-value-from-assigned-policies).
7. In case PoliciesManager receives *SDL.OnAllowSDLFunctionality* with *allowed=false* and with *device* param, PoliciesManager must record the named device (*device* param) as NOT consented in Local PT ("user_consent_records" -> *device* sub-section) and send *BC.ActivateApp* request with *level* param of the value from 'default_hmi' key of 'pre-DataConsent'section of Local PT to HMI.
8. In case PoliciesManager receives *SDL.OnAllowSDLFunctionality* with *allowed=true* and with *device* param from HMI, PoliciesManager must record the named device (*device* param) as consented in Local PT ("device_data" section \ subsection "user_consent_records" -> "device" sub-section).
9. PoliciesManager must store the User's consent for device (data consent) records in "device" subsection of "user_consent_records" subsection of `<device_identifier>` section of "device_data" section in Local PT on getting OnAllowSDLFunctionality from HMI.

PARAMETERS

NAME	TYPE	MANDATORY	DESCRIPTION
device	Common.DeviceInfo	false	If no device is specified permission counts for SDL functionality in general.
allowed	Boolean	true	Must be true if allowed
source	Common.ConsentSource	true	-

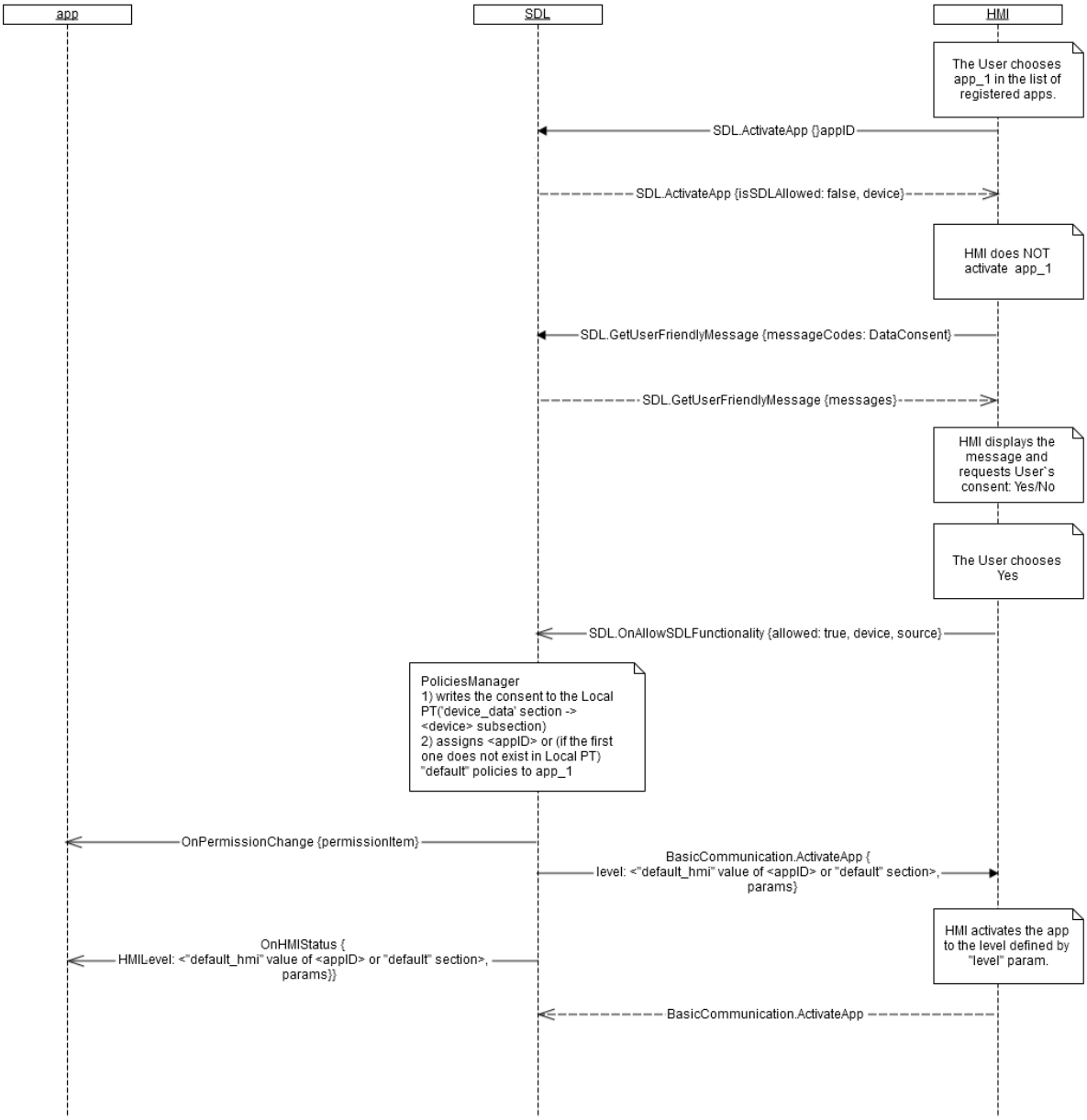
Sequence Diagrams

Preconditions:
- Device is connected to the HU.
- App_1 is running on this device and is registered with SDL.
- App_1 presents in the list of registered apps on HMI.

SEQUENCE DIAGRAM

The User consents the device.

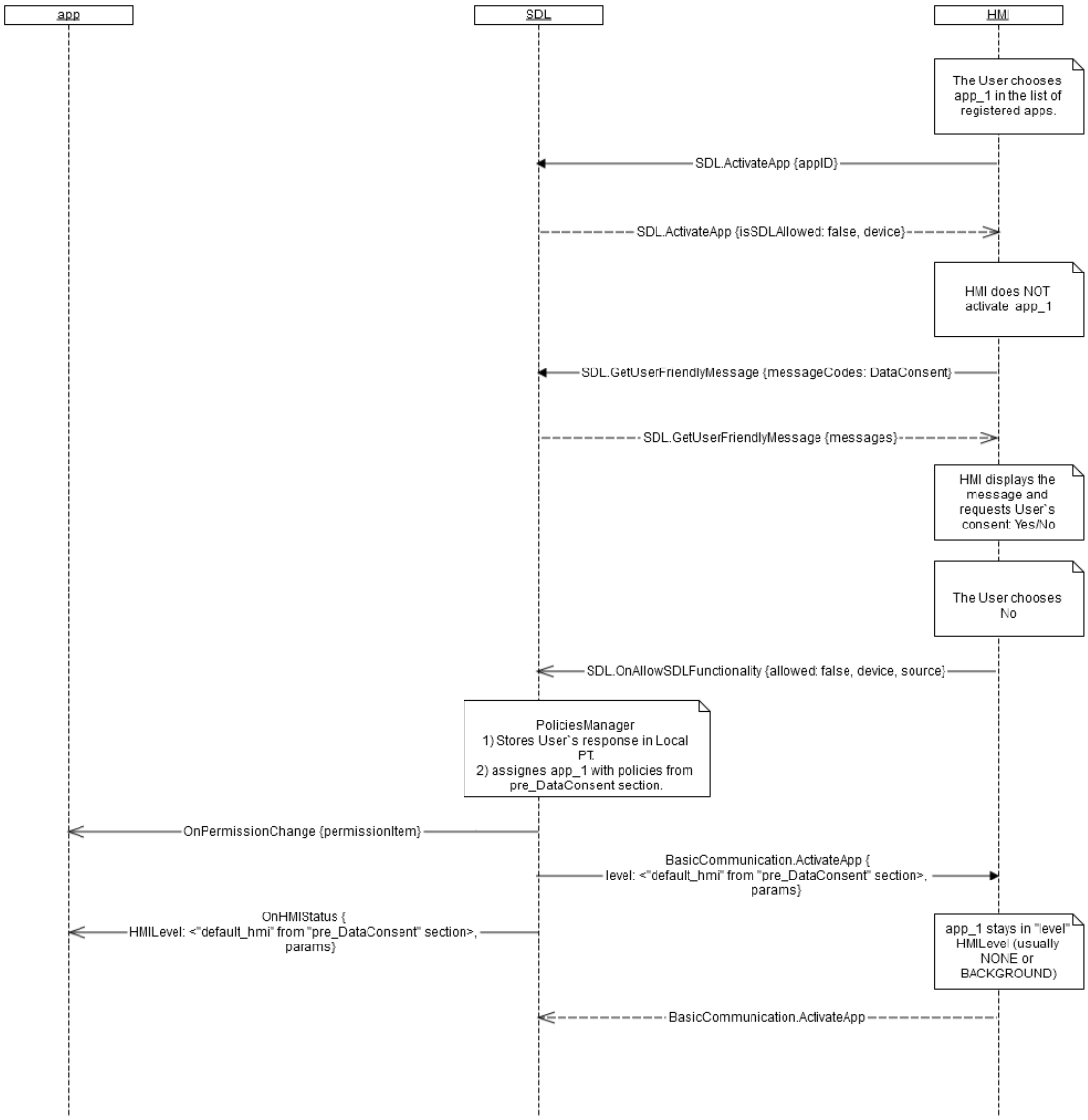
View Diagram



SEQUENCE DIAGRAM

The User does NOT consent the device.

View Diagram


app
SDL
HMI
The User chooses app_1 in the list of registered apps.
SDL.ActivateApp {appID}
SDL.ActivateApp {isSDLAllowed: false, device}
HMI does NOT activate app_1
SDL.GetUserFriendlyMessage {messageCodes: DataConsent}
SDL.GetUserFriendlyMessage {messages}
HMI displays the message and requests User`s consent: Yes/No
The User chooses No
SDL.OnAllowSDLFunctionality {allowed: false, device, source}
PoliciesManager
1) Stores User`s response in Local PT.
2) assignes app_1 with policies from pre_DataConsent section.
OnPermissionChange {permissionItem}
BasicCommunication.ActivateApp {
level: <"default_hmi" from "pre_DataConsent" section>,
params}
OnHMIStatus {
HMILevel: <"default_hmi" from "pre_DataConsent" section>,
params}
app_1 stays in "level" HMILevel (usually NONE or BACKGROUND)
BasicCommunication.ActivateApp

JSON EXAMPLE NOTIFICATION

```
{
   "jsonrpc" : "2.0",
   "method" : "SDL.OnAllowSDLFunctionality",
   "params" :
   {
      "deviceInfo":
      {
            "name" : “Mary`s Phone”,
            "id" : 8
      }

      “allowed” : true,
      “source” :”GUI”
   }
}
```

OnAppPermissionChanged

Type Notification
Sender SDL
Purpose

Inform HMI that permissions have changed for a specified application.

Notification from SDL to HMI occurs when application permissions were changed. If no permission specified means that application was dissallowed and has to be unregistered.

MUST

1. Request the list of permissions per app for which the consent is required (via `GetListOfPermissions`) when HMI gets the 'user-consent-required' notification per app from `SDL` (`SDL.OnAppPermissionChanged{appID, appPermissionsConsentNeeded: true}`).
2. Request Policies Manager for the appropriate user-friendly message from Local PT (`SDL.GetUserFriendlyMessage`) to commuticate about permissions changed when HMI gets the list of permissions that require User's consent.

It is app’s responsibility to inform the user to change the app’s settings if user consent is needed for proper app operation.
Any permissions that require user consent are not enabled until a user accepts them. If permissions do not require user consent, they can be applied immediately.

***SDL Note**:*

1. SDL sends the list of RequestTypes allowed by Policies via `OnAppPermissionChanged` API. In case HMI will use any of requestTypes disallowed by PolicyTable, SDL will ignore such notifications.
2. SDL PoliciesManager must send `OnAppPermissionChanged (appRevoked: true, appID)` and `BC.ActivateApp` (NONE) to HMI, in case the `<appID>` application is currently registered and in any HMILevel and in result of PTU `<appID>` gets "null" policies.
3. If app permissions were reduced after PTU, SDL sends `OnAppPermissionChanged (isAppPermissionsRevoked: true)` when app is

in FULL or LIMITTED. SDL doesn't notify HMI about revoked permissions twice.

4. In case the Local PT is successfully updated AND there are new application permissions that require User's consent, Policies Manager must:
   4.1. notify HMI about 'user-consent-required' via `SDL.OnAppPermissionChanged{appID, appPermissionsConsentNeeded: true}` per each application in FULL/LIMITED that lacks the User's permissions right after Policies Manager detects the user-unconsented permissions in Local PT.
   4.2. notify HMI about 'user-consent-required' `SDL.OnAppPermissionChanged{appID, appPermissionsConsentNeeded: true} after the related app is activated on HMI (in case the related app is now in BACKGROUND /NONE HMILevel)`.
5. In case the Local PT is successfully updated AND there are new permissions that require User's consent, Policies Manager must:
   5.1. notify HMI about 'user-consent-required' via `SDL.OnAppPermissionChanged{appID, appPermissionsConsentNeeded: true}` per each application that lacks the User's permissions.

5.2. provide the list of permissions that require User's consent upon request from HMI via `GetListOfPermissions(appID)`.

Notification

PARAMETERS

NAME	TYPE	MANDATORY	ADDITIONAL
appID	Integer	true	
isAppPermissionsRevoked	Boolean	false	
appRevokedPermissions	Common.PermissionItem	false	array: true minsize: 1 maxsize: 100
appRevoked	Boolean	false	
appPermissionsConsentNeeded	Boolean	false	
appUnauthorized	Boolean	false	
priority	Common.AppPriority	false	
requestType	Common.RequestType	false	array: true minsize: 0 maxsize: 100
requestSubType	String	false	array: true minsize: 0 maxsize: 100 maxlength: 100

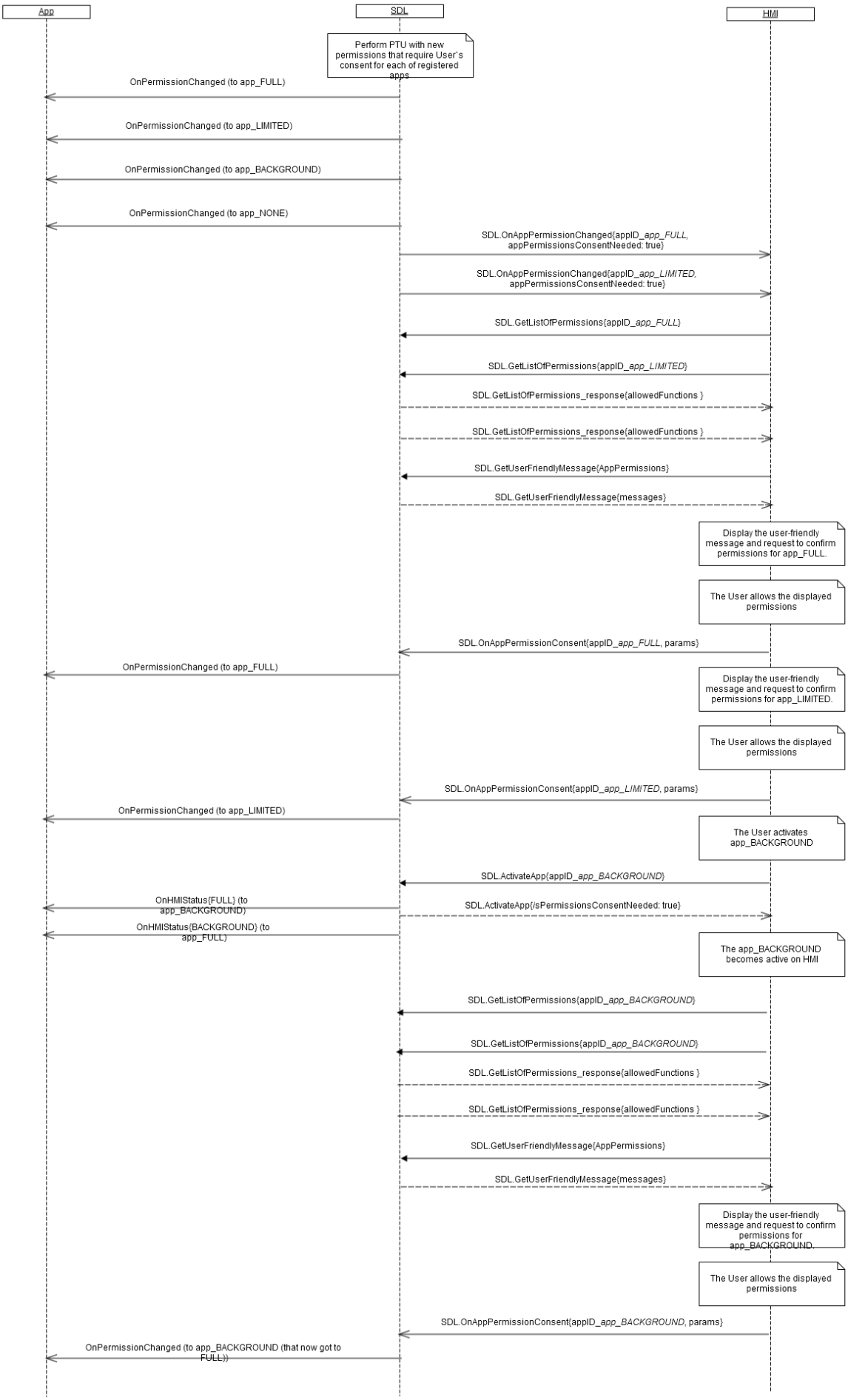
Sequence Diagrams

Preconditions to the sequence:
a) SDL and HMI are started;
b) Device is connected to the System (SDL/HU) and is consented by the User;
c) Four apps are registered with SDL and HMI ('name_HMILevel'): app_FULL, app_LIMITED, app_BACKGROUND, app_NONE.

SEQUENCE DIAGRAM

OnAppPermissionChanged with consent required

View Diagram


App
SDL
HMI
Perform PTU with new permissions that require User`s consent for each of registered apps
OnPermissionChanged (to app_FULL)
OnPermissionChanged (to app_LIMITED)
OnPermissionChanged (to app_BACKGROUND)
OnPermissionChanged (to app_NONE)
SDL.OnAppPermissionChanged{appID_app_FULL, appPermissionsConsentNeeded: true}
SDL.OnAppPermissionChanged{appID_app_LIMITED, appPermissionsConsentNeeded: true}
SDL.GetListOfPermissions{appID_app_FULL}
SDL.GetListOfPermissions{appID_app_LIMITED}
SDL.GetListOfPermissions_response{allowedFunctions }
SDL.GetListOfPermissions_response{allowedFunctions }
SDL.GetUserFriendlyMessage{AppPermissions}
SDL.GetUserFriendlyMessage{messages}
Display the user-friendly message and request to confirm permissions for app_FULL.
The User allows the displayed permissions
SDL.OnAppPermissionConsent{appID_app_FULL, params}
OnPermissionChanged (to app_FULL)
Display the user-friendly message and request to confirm permissions for app_LIMITED.
The User allows the displayed permissions
SDL.OnAppPermissionConsent{appID_app_LIMITED, params}
OnPermissionChanged (to app_LIMITED)
The User activates app_BACKGROUND
SDL.ActivateApp{appID_app_BACKGROUND}
OnHMIStatus{FULL} (to app_BACKGROUND)
SDL.ActivateApp{isPermissionsConsentNeeded: true}
OnHMIStatus{BACKGROUND} (to app_FULL)
The app_BACKGROUND becomes active on HMI
SDL.GetListOfPermissions{appID_app_BACKGROUND}
SDL.GetListOfPermissions{appID_app_BACKGROUND}
SDL.GetListOfPermissions_response{allowedFunctions }
SDL.GetListOfPermissions_response{allowedFunctions }
SDL.GetUserFriendlyMessage{AppPermissions}
SDL.GetUserFriendlyMessage{messages}
Display the user-friendly message and request to confirm permissions for app_BACKGROUND.
The User allows the displayed permissions
SDL.OnAppPermissionConsent{appID_app_BACKGROUND, params}
OnPermissionChanged (to app_BACKGROUND (that now got to FULL))

JSON EXAMPLE NOTIFICATION

```
{
 "jsonrpc" : "2.0",
 "method" : "SDL.OnAppPermissionChanged",
 "params" :
 {
  "appID" : 65674,

 }
}
```

OnAppPermissionConsent

Notification

Type Notification
Sender HMI
Purpose Inform about the User permission changes for some functionality for the named application.

Initiated by HMI for specifying the allowance for the application to perform some functionality.The notification informs Policy Manager about some changes in application permissions, that effect application behavior on HMI.

MUST

1) send `OnAppPermissionConsent` when User answers to prompt about app's permisstions consent.
2) send `OnAppPermissionConsent` when User enters settings menu and allows/disallows app's permissions.
3) use the pair of values id<->name in PermissionItem structure which were obtained via `GetListOfPermissions` response
4) send `OnAppPermissionConsent` when User changes `ExternalConsentStatus`.

SDL information:
PoliciesManager applies the changes to all applications in case `OnAppPermissionConsent` is received without `<appID>` parameter.
PoliciesManager applies the changes received via `OnAppPermissionConsent` according to its internal rules (update appropriate application permissions sections in the policies database etc).

NOTE

a) SDL ignores all invalid notifications which come from HMI (invalid JSON, invalid data types/bounds etc).
b) `ExternalConsentStatus` either user_dissalows or user_allows applications functional groupings depending on predefined settings in policy table.
c) SDL uses `OnAppPermissionConsent` value (ON/OFF) received from HMI through ignition cycles until this value is changed by corresponding notification from HMI.

PARAMETERS

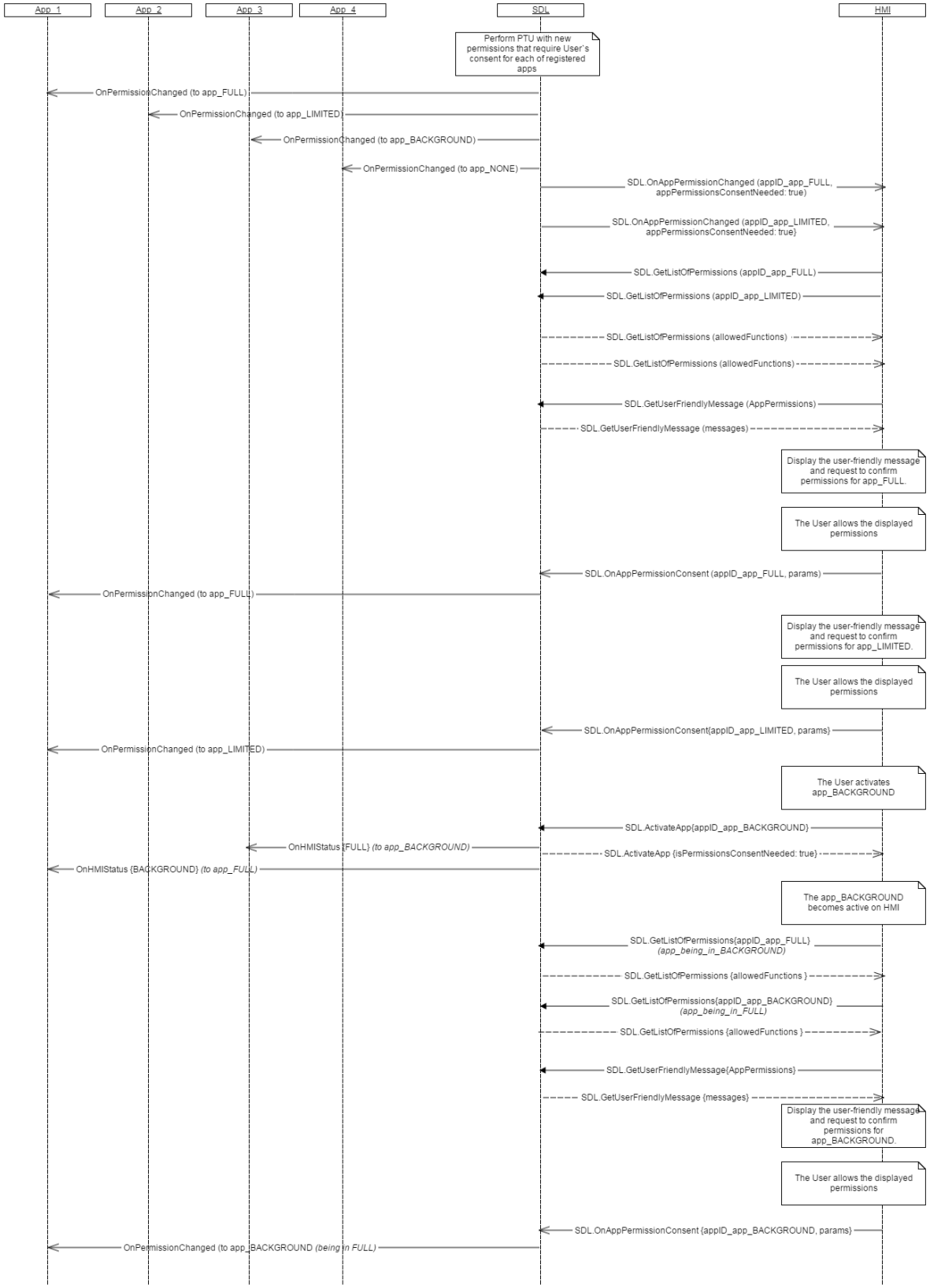
NAME	TYPE	MANDATORY	ADDITIONAL
appID	Integer	false	
consentedFunctions	Common.PermissionItem	false	array: true minsize: 1 maxsize: 100
ExternalConsentStatus	Common.ExternalConsentStatus	false	array: true minsize: 1 maxsize: 100
source	Common.ConsentSource	true	

Sequence Diagrams

SEQUENCE DIAGRAM

OnAppPermissionConsent

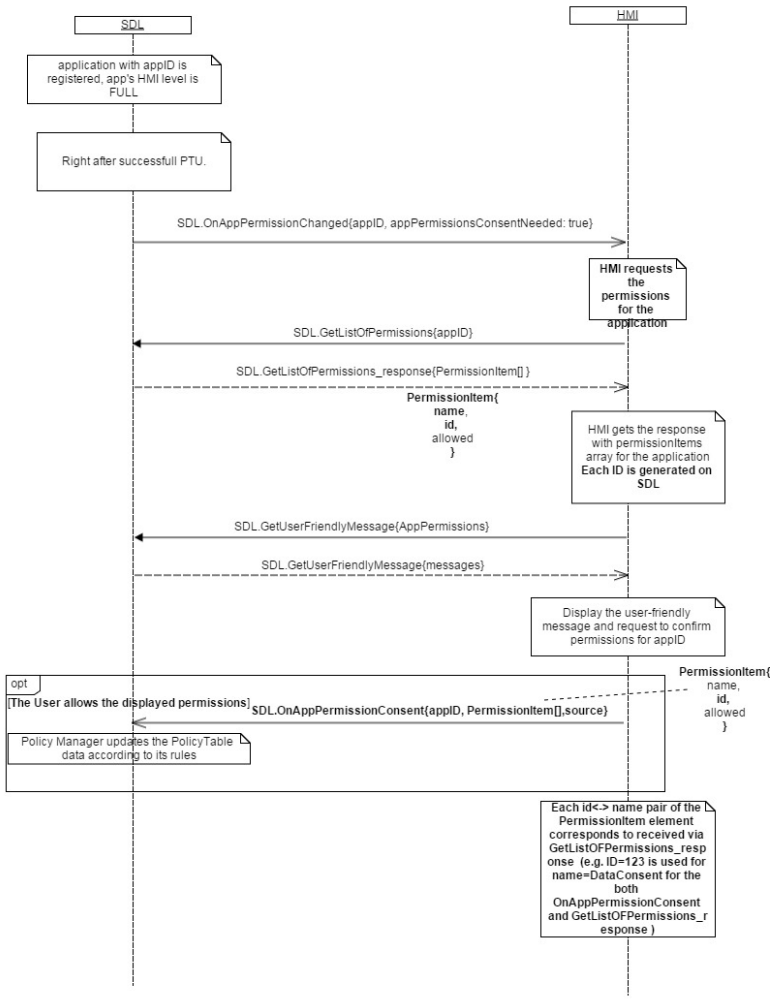
View Diagram



SEQUENCE DIAGRAM

OnAppPermissionConsent (id<->name dependency)

View Diagram



JSON EXAMPLE NOTIFICATION

```
{
    "jsonrpc" : "2.0",
    "method" : "SDL.OnAppPermissionConsent",
    "params" :
    {
    "appID":13759,
    "consentedFunctions":[
     {"allowed":false,
      "id":4734356,
      "name":"DrivingCharacteristics"
     }],
    "source":"GUI"
    }}
```

OnDeviceStateChanged

Type Notification
Sender HMI
Purpose Inform about BLUETOOTH device state changed

SDL PoliciesManager needs this notification for updating the data in policies database. Information about unpaired device will cause SDL PoliciesManager to remove all data related to it from the database.

MUST

Notify SDL when the state of bluetooth paired device has been changed.

NOTE

SDL ignores all invalid notifications which come from HMI (invalid JSON, invalid data types/bounds etc).
Unpairing of the device is NOT a trigger for new PTU sequence.

PARAMETERS

NAME	TYPE	MANDATORY	ADDITIONAL
deviceState	Common.DeviceState	true	-
deviceInternalId	String	true	minlength: 0 maxlength: 500
deviceId	Common.DeviceInfo	false	-

JSON EXAMPLE NOTIFICATION

```
{
   "jsonrpc" : "2.0",
   "method" : "SDL.OnDeviceStateChanged",
   "params" :
   {
      "deviceInternalId":"0017CAF56A51",
    "deviceState":"UNPAIRED"  }
}
```

OnPolicyUpdate

Type Notification
Sender HMI
Purpose Notifies SDL to supply a new "PolicyUpdate" request with more recent snapshot data

MUST

- Send *OnPolicyUpdate* to SDL if it is required to get Policy Table Update (by any specific HMI workflow reason).
- Continue operating according to Policies workflow in case SDL started Policy Update process as a result of BC.OnPolicyUpdate

NOTE

HMI should send *SDL.OnPolicyUpdate* in case the new PT Snapshot is required from PoliciesManager.
Currently this notification is not used in any HMI workflows, but in case of necessity, HMI may use *OnPolicyUpdate* for getting the newest Snapshot (for example, during retry sequence or by user request).

NOTE

- SDL ignores all invalid notifications which come from HMI (invalid JSON, invalid data types/bounds etc).
- In case SDL receives *SDL.OnPolicyUpdate* notification from HMI, SDL PoliciesManager must start the procedure of Policy Table Update (that is, create PT Snapshot, send it to HMI for encryption, and etc. what is defined by related requirements).

PARAMETERS

This RPC has no additional parameter requirements.

Sequence Diagrams

SEQUENCE DIAGRAM

Policy Table Update flow

View Diagram

JSON EXAMPLE NOTIFICATION

```
{
   "jsonrpc" : "2.0",
   "method" : "SDL.OnPolicyUpdate"
}
```

OnReceivedPolicyUpdate

Type Notification
Sender

HMI

Purpose

Trigger SDL to merge the Updated Policy Table to the Local Policy Table

Notification

MUST

- Send *SDL.OnReceivedPolicyUpdate* notification to SDL after HMI finalized processing the updated Policy Table delivered via *BC.SystemRequest* (for example, after decrypting it *in case* and by the scheme required by Policies Server).
- Decrypt the PTU file received via SystemRequest.
- Notify SDL on successful decryption and provide the path to decrypted PTU file.

NOTE

1. *SDL.OnReceivedPolicyUpdate* dependencies:
2. SDL expects *SDL.OnReceivedPolicyUpdate only in case* it's built with `"-DEXTENDED_POLICY: PROPRIETARY" flag or without this flag` and `-DEXTENDED_POLICY: EXTERNAL_PROPRIETARY` flag. *Otherwise* SDL handles the entire PTU flow by itself.
3. SDL will not use Updated PT until notified by HMI.
4. After getting *OnReceivedPolicyUpdate (policyFile)* from HMI, *SDL must* stop timeout started by *OnSystemRequest* and validate the Policy Table Update (policyFile) of optional, required, or omitted:
   - validation must reject Policy Table updates if it include fields with a status of ‘omitted.’
   - validation must reject Policy Table update if it does not include fields with a status of ‘required’.
5. In case section with required status "optional/omitted" is omitted in Updated PT, and field of this section is marked as required, the validation of the mentioned field is not "required" (i.e. policy table must be considered as valid).

PARAMETERS

NAME	TYPE	MANDATORY	ADDITIONAL
policyFile	String	true	minlength: 1 maxlength: 255

Sequence Diagrams

SEQUENCE DIAGRAM

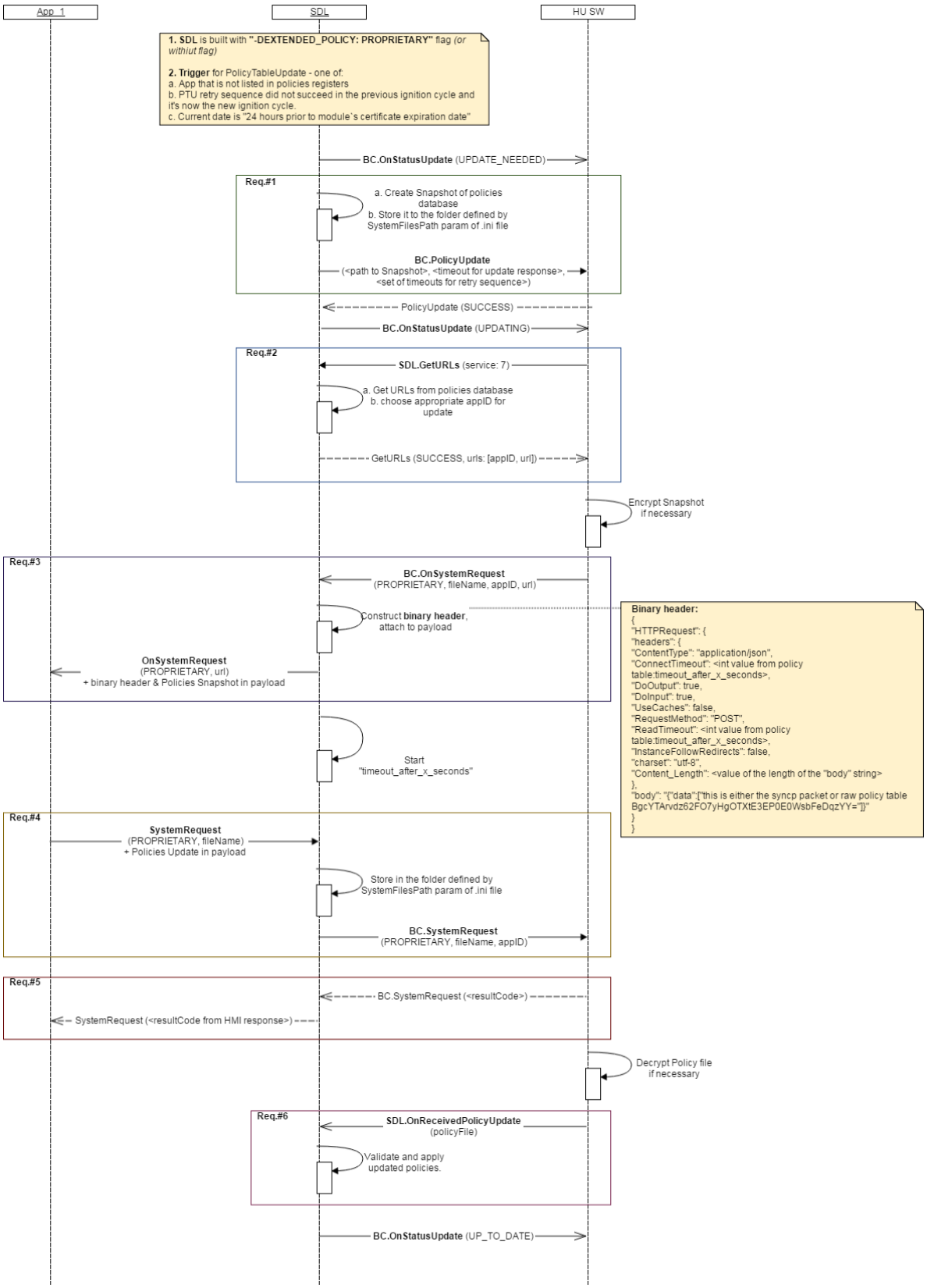
SDL.OnReceivedPolicyUpdate in EXTERNAL_PROPRIETARY flow

View Diagram

SEQUENCE DIAGRAM

SDL.OnReceivedPolicyUpdate in "Proprietary" Policy Table Update Flow

View Diagram


App_1
SDL
HU SW
1. SDL is built with "-DEXTENDED_POLICY: PROPRIETARY" flag (or withiut flag)
2. Trigger for PolicyTableUpdate - one of:
a. App that is not listed in policies registers
b. PTU retry sequence did not succeed in the previous ignition cycle and it's now the new ignition cycle.
c. Current date is "24 hours prior to module`s certificate expiration date"
BC.OnStatusUpdate (UPDATE_NEEDED)
Req.#1
a. Create Snapshot of policies database
b. Store it to the folder defined by SystemFilesPath param of .ini file
BC.PolicyUpdate
(<path to Snapshot>, <timeout for update response>, <set of timeouts for retry sequence>)
PolicyUpdate (SUCCESS)
BC.OnStatusUpdate (UPDATING)
Req.#2
SDL.GetURLs (service: 7)
a. Get URLs from policies database
b. choose appropriate appID for update
GetURLs (SUCCESS, urls: [appID, url])
Encrypt Snapshot if necessary
Req.#3
BC.OnSystemRequest
(PROPRIETARY, fileName, appID, url)
Construct binary header, attach to payload
OnSystemRequest
(PROPRIETARY, url)
+ binary header & Policies Snapshot in payload
Binary header:
{
"HTTPRequest": {
"headers": {
"ContentType": "application/json",
"ConnectTimeout": <int value from policy table:timeout_after_x_seconds>,
"DoOutput": true,
"DoInput": true,
"UseCaches": false,
"RequestMethod": "POST",
"ReadTimeout": <int value from policy table:timeout_after_x_seconds>,
"InstanceFollowRedirects": false,
"charset": "utf-8",
"Content_Length": <value of the length of the "body" string>
},
"body": "{"data":["this is either the syncp packet or raw policy table BgcYTArvdz62FO7yHgOTXtE3EP0E0WsbFeDqzYY="]}"
}
}
Start "timeout_after_x_seconds"
Req.#4
SystemRequest
(PROPRIETARY, fileName)
+ Policies Update in payload
Store in the folder defined by SystemFilesPath param of .ini file
BC.SystemRequest
(PROPRIETARY, fileName, appID)
Req.#5
BC.SystemRequest (<resultCode>)
SystemRequest (<resultCode from HMI response>)
Decrypt Policy file if necessary
Req.#6
SDL.OnReceivedPolicyUpdate
(policyFile)
Validate and apply updated policies.
BC.OnStatusUpdate (UP_TO_DATE)

JSON EXAMPLE NOTIFICATION

```
{
   "id" : 176,
   "jsonrpc" : "2.0",
   "method" : "SDL.GetURLS",
   "params" :
   {
      "policyfile" : "/fs/sharedFolder/ptu.json"
   }
}
```

OnSDLConsentNeeded

Type Notification
Sender SDL
Purpose Send from SDL to HMI to notify that data consent is needed for device

The notification is applicable only for the case when the device is consented by the user and the user requests Policy Table update from HMI.

MUST

1) Initiate the sequence of device data consent on getting OnSDLConsentNeeded.
2) Upon getting *OnSDLConsentNeeded*, request *SDL.GetUserFriendlyMessages* from SDL.
3) Display the 'device consent' dialog after getting *SDL.GetUserFriendlyMessages_response* from SDL.
4) Send *OnAllowSDLFunctionality* (true or false) after and depending on User's choice on HMI.

PARAMETERS

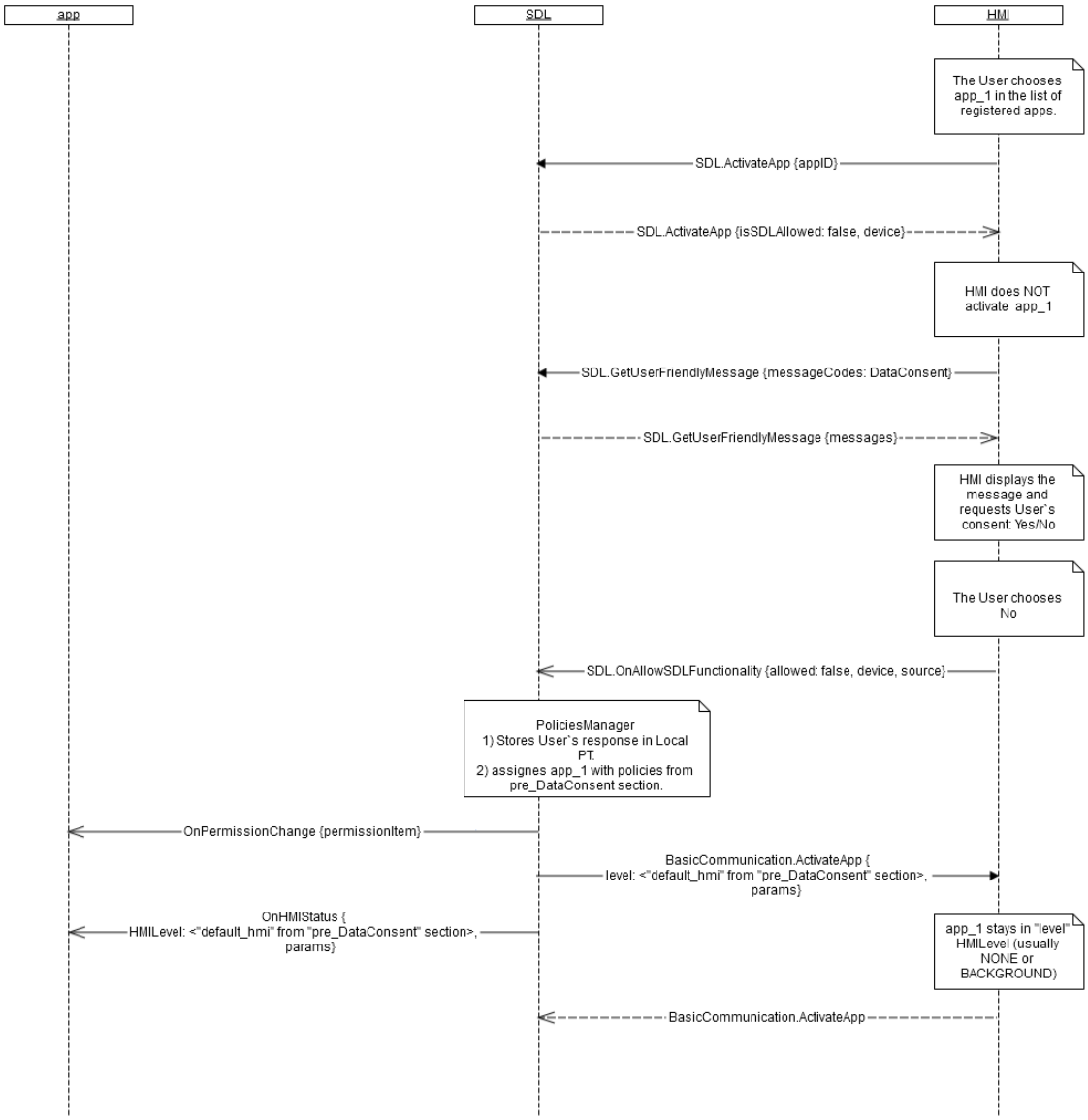
NAME	TYPE	MANDATORY	ADDITIONAL
device	Common.DeviceInfo	true	

Sequence Diagrams

SEQUENCE DIAGRAM

The User does NOT consent the device

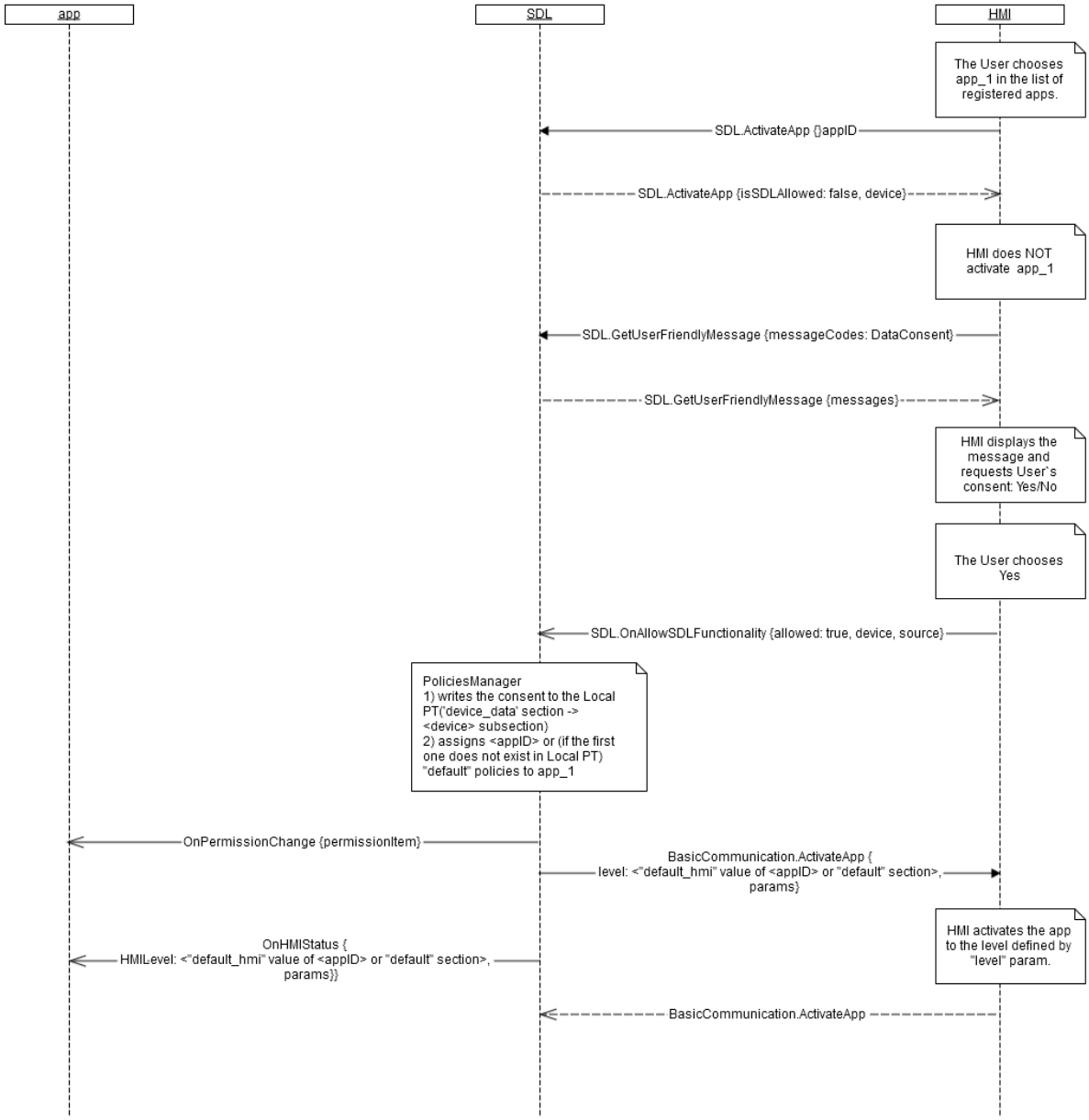
View Diagram



SEQUENCE DIAGRAM

The User consents the device

View Diagram


app
SDL
HMI
The User chooses app_1 in the list of registered apps.
SDL.ActivateApp {}appID
SDL.ActivateApp {isSDLAllowed: false, device}
HMI does NOT activate app_1
SDL.GetUserFriendlyMessage {messageCodes: DataConsent}
SDL.GetUserFriendlyMessage {messages}
HMI displays the message and requests User`s consent: Yes/No
The User chooses Yes
SDL.OnAllowSDLFunctionality {allowed: true, device, source}
PoliciesManager
1) writes the consent to the Local PT('device_data' section -> <device> subsection)
2) assigns <appID> or (if the first one does not exist in Local PT) "default" policies to app_1
OnPermissionChange {permissionItem}
BasicCommunication.ActivateApp {
level: <"default_hmi" value of <appID> or "default" section>,
params}
HMI activates the app to the level defined by "level" param.
OnHMIStatus {
HMILevel: <"default_hmi" value of <appID> or "default" section>,
params}}
BasicCommunication.ActivateApp

JSON EXAMPLE NOTIFICATION

```
{
   "jsonrpc" : "2.0",
   "method" : "SDL.OnSDLConsentNeeded",
   "params" :
      {
      "deviceInfo" :
      {
            "name" : “Mary`s Phone”,
            "id" : 8
      }
   }
}
```

OnStatusUpdate

Type Notification
Sender SDL
Purpose Inform the HMI about the status of a Policy Table Update (PTU).

Notification

MAY

Inform the User about the status of PTU by updating the appropriate UI screens.

NOTE

SDL operates with the following PTU statuses:
UPDATE_NEEDED - one of the triggers for PTU occurs (see BC.PolicyUpdate)
UPDATING - SDL starts the PTU flow by creating the Snapshot of Local Policy Table
* UP_TO_DATE - SDL merges the Updated Policy Table to the Local one

PARAMETERS

NAME	TYPE	MANDATORY	ADDITIONAL
status	Common.UpdateResult	true	

JSON EXAMPLE NOTIFICATION

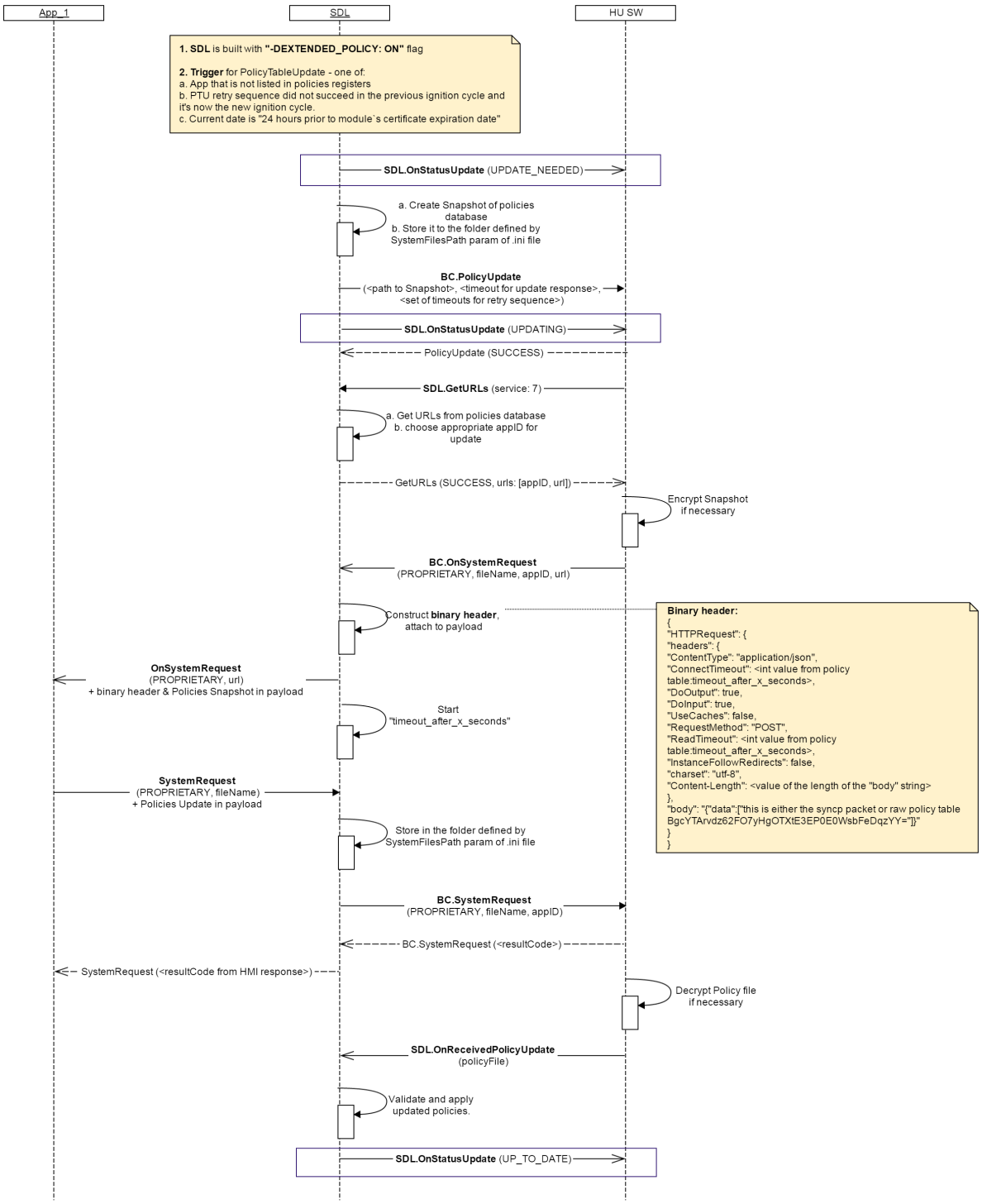
```
{
 "jsonrpc" : "2.0",
 "method" : "SDL.OnStatusUpdate",
 "params" :
 {
  "status" : "UPDATE_NEEDED"
 }
}
```

Sequence Diagrams

SEQUENCE DIAGRAM

SDL.OnStatusUpdate in "Proprietary" Policy Table Update Flow

View Diagram


App_1
SDL
HU SW
1. SDL is built with "-DEXTENDED_POLICY: ON" flag
2. Trigger for PolicyTableUpdate - one of:
a. App that is not listed in policies registers
b. PTU retry sequence did not succeed in the previous ignition cycle and it's now the new ignition cycle.
c. Current date is "24 hours prior to module`s certificate expiration date"
SDL.OnStatusUpdate (UPDATE_NEEDED)
a. Create Snapshot of policies database
b. Store it to the folder defined by SystemFilesPath param of .ini file
BC.PolicyUpdate
(<path to Snapshot>, <timeout for update response>, <set of timeouts for retry sequence>)
SDL.OnStatusUpdate (UPDATING)
PolicyUpdate (SUCCESS)
SDL.GetURLs (service: 7)
a. Get URLs from policies database
b. choose appropriate appID for update
GetURLs (SUCCESS, urls: [appID, url])
Encrypt Snapshot if necessary
BC.OnSystemRequest
(PROPRIETARY, fileName, appID, url)
Construct binary header, attach to payload
Binary header:
{
"HTTPRequest": {
"headers": {
"ContentType": "application/json",
"ConnectTimeout": <int value from policy table:timeout_after_x_seconds>,
"DoOutput": true,
"DoInput": true,
"UseCaches": false,
"RequestMethod": "POST",
"ReadTimeout": <int value from policy table:timeout_after_x_seconds>,
"InstanceFollowRedirects": false,
"charset": "utf-8",
"Content-Length": <value of the length of the "body" string>
},
"body": "{"data":["this is either the syncp packet or raw policy table
BgcYTArvdz62FO7yHgOTXtE3EP0E0WsbFeDqzYY="]}"
}
}
OnSystemRequest
(PROPRIETARY, url)
+ binary header & Policies Snapshot in payload
Start "timeout_after_x_seconds"
SystemRequest
(PROPRIETARY, fileName)
+ Policies Update in payload
Store in the folder defined by SystemFilesPath param of .ini file
BC.SystemRequest
(PROPRIETARY, fileName, appID)
BC.SystemRequest (<resultCode>)
SystemRequest (<resultCode from HMI response>)
Decrypt Policy file if necessary
SDL.OnReceivedPolicyUpdate
(policyFile)
Validate and apply updated policies.
SDL.OnStatusUpdate (UP_TO_DATE)

OnSystemError

Notification

PARAMETERS

NAME	TYPE	MANDATORY	ADDITIONAL
error	Common.SystemError	true	

JSON EXAMPLE NOTIFICATION

UpdateSDL

Notification

Type

Function

Sender HMI
Purpose Update the Policy Table request

REQUEST

The request notifies PolicyManager about Policy Table Update is requested by User on HMI.
The below describes the behavior related to EXTERNAL_PROPRIETARY Policy flow only.

MUST

1) Send UpdateSDL request to SDL in case the user requests Policy Table update on HMI.
2) After UpdateSDL response follow the process of Policy Update according to “result” received.

NOTE

According to result sent to HMI in UpdateSDL response, SDL will follow Policy Update process:
1) UP_TO_DATE – Policy Table is now up to date, but anyway SDL starts a new policy update cycle because of user request (e.i. UpdateSDL).
2) UPDATING – Policy Update process is now in already progress, anyway when currently active update process will be finished, SDL must start PT Update from the beginning.
3) UPDATE_NEEDED – Policy Table is not up to date, update process must be started by SDL.
PoliciesManager must initiate the PT Update sequence (that is, PT Exchange) upon User`s request delivered to SDL via SDL.UpdateSDL() from HMI and provide a response on a request with current PTU status to HMI.
Information: PROPRIETARY Policy flow -> SDL must change the flag from "UPDATE_NEEDED" to "UPDATING" right after *BC.PolicyUpdate* with SnapshotPT is sent to HMI.

PARAMETERS

This RPC has no additional parameter requirements.

RESPONSE

PARAMETERS

NAME	TYPE	MANDATORY	DESCRIPTION
result	Common.UpdateResult	true	Specify result: no update needed, update was successful/ unsuccessful, etc.

Sequence Diagrams

SEQUENCE DIAGRAM

UpdateSDL UP_TO_DATE - EXTERNAL_PROPRIETARY Policy Table Update Flow

View Diagram

SEQUENCE DIAGRAM

UpdateSDL UPDATING - EXTERNAL_PROPRIETARY Policy Table Update Flow

View Diagram

SEQUENCE DIAGRAM

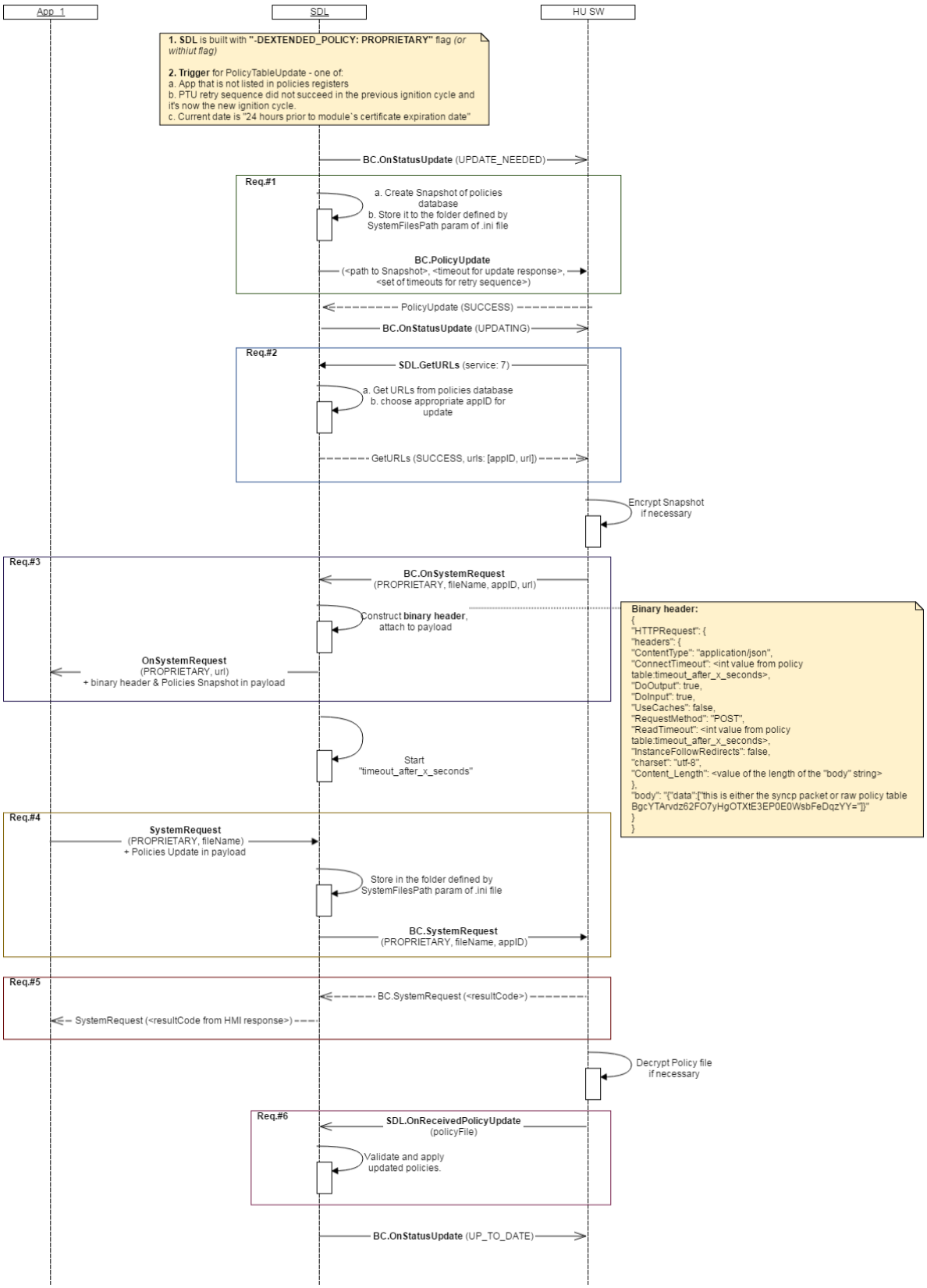
UpdateSDL UPDATE_NEEDED - EXTERNAL_PROPRIETARY Policy Table Update Flow

View Diagram

SEQUENCE DIAGRAM

UpdateSDL - PROPRIETARY Policy Table Update Flow

View Diagram


App_1
SDL
HU SW
1. SDL is built with "-DEXTENDED_POLICY: PROPRIETARY" flag (or withiut flag)
2. Trigger for PolicyTableUpdate - one of:
a. App that is not listed in policies registers
b. PTU retry sequence did not succeed in the previous ignition cycle and it's now the new ignition cycle.
c. Current date is "24 hours prior to module`s certificate expiration date"
BC.OnStatusUpdate (UPDATE_NEEDED)
Req.#1
a. Create Snapshot of policies database
b. Store it to the folder defined by SystemFilesPath param of .ini file
BC.PolicyUpdate
(<path to Snapshot>, <timeout for update response>, <set of timeouts for retry sequence>)
PolicyUpdate (SUCCESS)
BC.OnStatusUpdate (UPDATING)
Req.#2
SDL.GetURLs (service: 7)
a. Get URLs from policies database
b. choose appropriate appID for update
GetURLs (SUCCESS, urls: [appID, url])
Encrypt Snapshot if necessary
Req.#3
BC.OnSystemRequest
(PROPRIETARY, fileName, appID, url)
Construct binary header, attach to payload
OnSystemRequest
(PROPRIETARY, url)
+ binary header & Policies Snapshot in payload
Binary header:
{
"HTTPRequest": {
"headers": {
"ContentType": "application/json",
"ConnectTimeout": <int value from policy table:timeout_after_x_seconds>,
"DoOutput": true,
"DoInput": true,
"UseCaches": false,
"RequestMethod": "POST",
"ReadTimeout": <int value from policy table:timeout_after_x_seconds>,
"InstanceFollowRedirects": false,
"charset": "utf-8",
"Content_Length": <value of the length of the "body" string>
},
"body": "{"data":["this is either the syncp packet or raw policy table BgcYTArvdz62FO7yHgOTXtE3EP0E0WsbFeDqzYY="]}"
}
}
Start "timeout_after_x_seconds"
Req.#4
SystemRequest
(PROPRIETARY, fileName)
+ Policies Update in payload
Store in the folder defined by SystemFilesPath param of .ini file
BC.SystemRequest
(PROPRIETARY, fileName, appID)
Req.#5
BC.SystemRequest (<resultCode>)
SystemRequest (<resultCode from HMI response>)
Decrypt Policy file if necessary
Req.#6
SDL.OnReceivedPolicyUpdate
(policyFile)
Validate and apply updated policies.
BC.OnStatusUpdate (UP_TO_DATE)

Example Request

```
{
   "id" : 543,
   "jsonrpc" : "2.0",
   "method" : "SDL.UpdateSDL"
}
```

Example Response

```
{
   "id" : 543,
   "jsonrpc" : "2.0",
   "result" :
   {
      “result” : “UPDATING”
      "code" : 0,
      "method" : "SDL.UpdateSDL"
   }
}
```

Example Error

```
{
   "id" : 543,
   "jsonrpc" : "2.0",
   "error" :
   {
      "code" : 22,
      "message" : "The unknown error has occurred",
      "data" :
      {
         "method" : "SDL.UpdateSDL"
      }
   }
}
```

ChangeRegistration

Type Function
Sender SDL
Purpose Change the language of the TTS component.

REQUEST

PARAMETERS

NAME	TYPE	MANDATORY	ADDITIONAL
ttsName	Common.TTSChunk	false	array: true minsize: 1 maxsize: 100
language	Common.Language	true	
appID	Integer	true	

RESPONSE

PARAMETERS

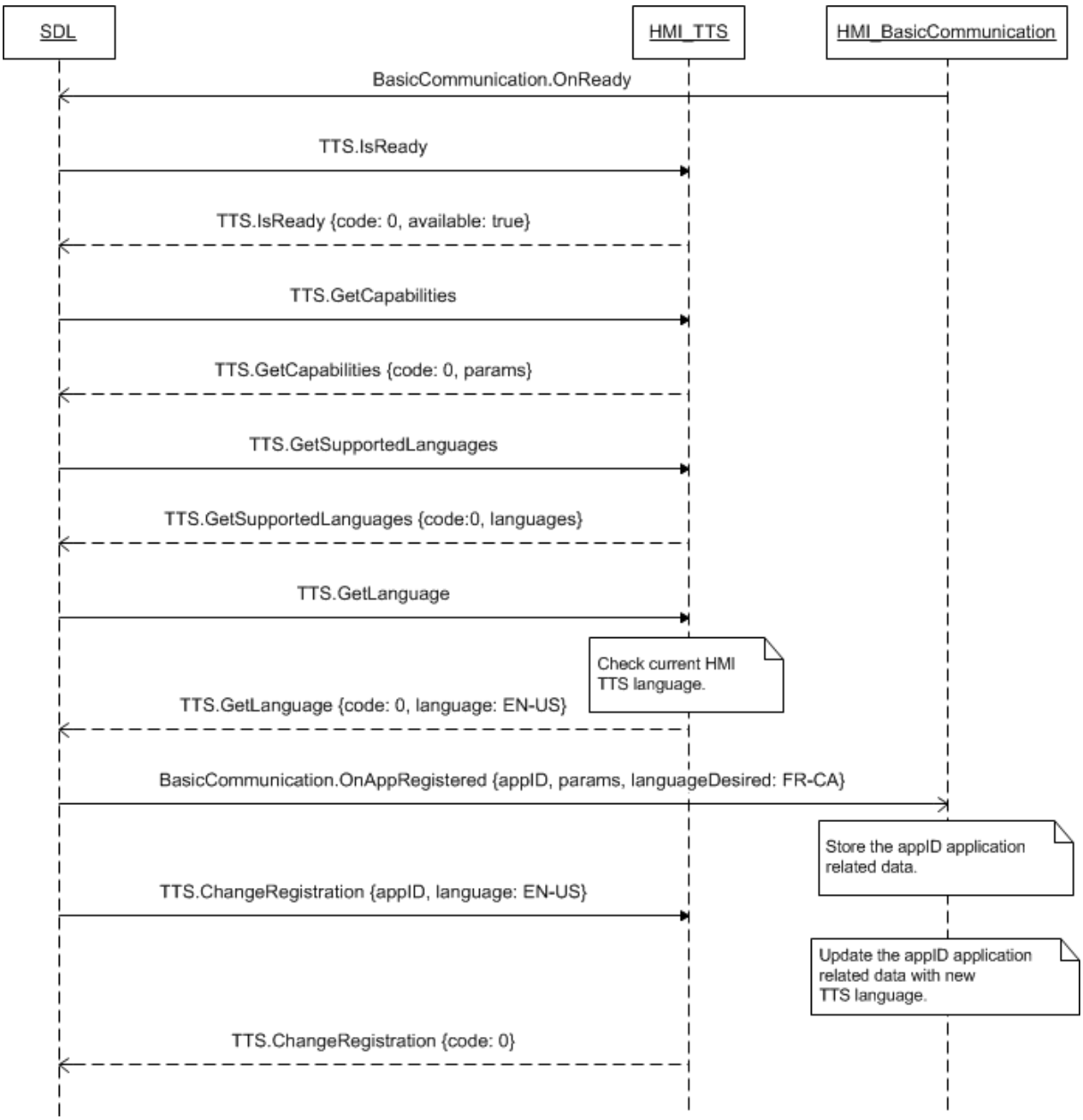
This RPC has no additional parameter requirements

Sequence Diagrams

SEQUENCE DIAGRAM

ChangeRegistration after OnAppRegistered

View Diagram



Example Request

```
{
 "id" : 206,
 "jsonrpc" : "2.0",
 "method" : "TTS.ChangeRegistration",
 "params" :
 {
  "language" : DE-DE,
  "appID" : 65539
 }
}
```

Example Response


```
{
 "id" : 206,
 "jsonrpc" : "2.0",
 "result" :
 {
  "code" : 0,
  "method" : "TTS.ChangeRegistration"
 }
}
```


Example Error


```
{
 "id" : 206,
 "jsonrpc" : "2.0",
 "error" :
 {
  "code" : 22,
  "message" : "Unknown error occurred",
  "data" :
  {
   "method" : "TTS.ChangeRegistration"
  }
 }
}
```

GetCapabilities

Type Function
Sender SDL
Purpose Inform SDL of the TTS capabilities of the vehicle.

REQUEST

PARAMETERS

NAME	TYPE	MANDATORY	ADDITIONAL

RESPONSE

PARAMETERS

NAME	TYPE	MANDATORY	ADDITIONAL
speechCapabilities	Common.SpeechCapabilities	true	array: true minsize: 1 maxsize: 5
prerecordedSpeechCapabilities	Common.PrerecordedSpeech	true	array: true minsize: 1 maxsize: 5

NOTE

Description of possible SpeechCapabilities

TTS is performed by the HMI through:

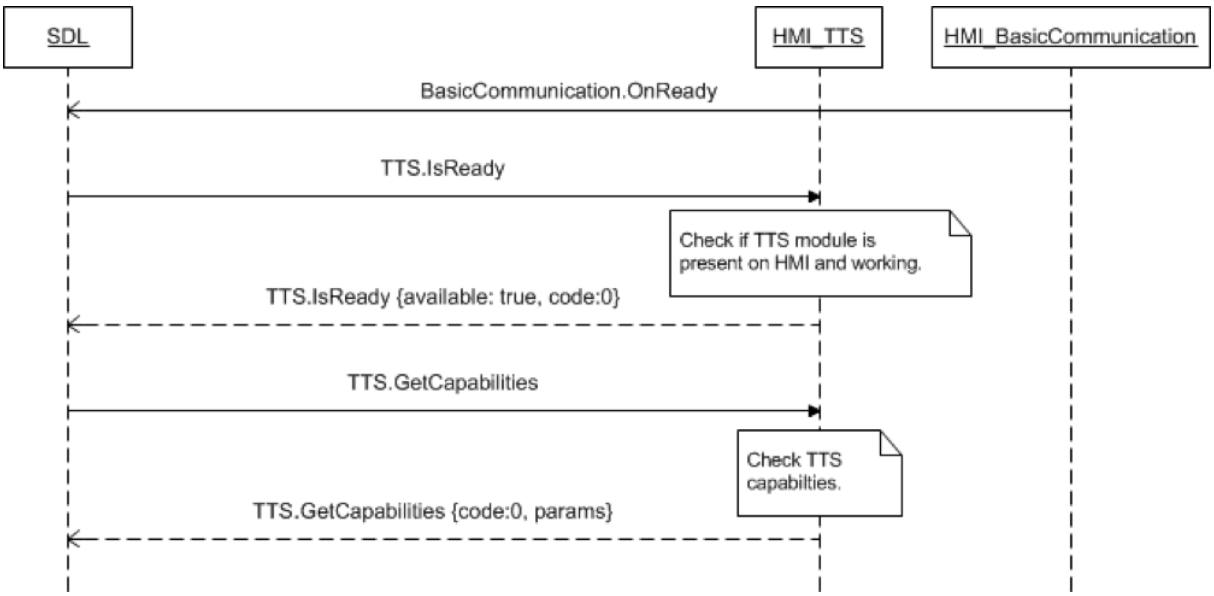
1. TEXT - Plain text format. TTSChunks with this type provide plain text to be spoken in the `text` field
2. SAPI_PHONEMES - Microsoft Speech API Format. TTSChunks with this type provide a group of phonemes with this format in the `text` field
3. LHPLUS_PHONEMES - LH+ Phoneme format. TTSChunks with this type provide a group of phonemes with this format in the `text` field
4. PRE_RECORDED - Prerecorded sounds. TTSChunks with this type provide a value from Common.PrerecordedSpeech in the `text` field
5. SILENCE - Definite amount of time. TTSChunks with this type provide amount of time HMI must keep silence.
6. FILE - Uploaded audio files. TTSChunks with this type provide a filename pointing to an audio file previously uploaded through a PutFile RPC in the `text` field
    - These audio files can be in one of the following file types/ specifications, all must be supported in order for the HMI to support the `FILE` option:
        - AUDIO_WAVE
            - Sample Rate: 16000 Hz
            - Bit Depth: 16
            - Mono Channel (1 Channel)
        - AUDIO_MP3
            - Sample Rate: 44100 Hz
            - Bit Depth: 8 - 320
            - Mono Channel (1 Channel)
        - AUDIO_AAC
            - Sample Rate: 44100 Hz
            - Bit Depth: 8 - 320
            - Mono Channel (1 Channel)

Sequence Diagrams

SEQUENCE DIAGRAM

GetCapabilities

View Diagram



Example Request

```
{
  "id" : 13,
  "jsonrpc" : "2.0",
  "method" : "TTS.GetCapabilities"
}
```

Example Response


```
{
 "id" : 13,
 "jsonrpc" : "2.0",
 "result" :
 {
  "capabilities" : [TEXT],
  "prerecordedSpeechCapabilities" : [HELP_JINGLE, INITIAL_JINGLE,
LISTEN_JINGLE, POSITIVE_JINGLE, NEGATIVE_JINGLE],
  "code" : 0,
  "method" : "TTS.GetCapabilities"
 }
}
```


Example Error


```
{
 "id" : 28,
 "jsonrpc" : "2.0",
 "error" :
 {
  "code" : 11,
  "message" : "The data sent is invalid",
  "data" :
  {
   "method" : "TTS.GetCapabilities"
  }
 }
}
```

GetLanguage

Type Function
Sender SDL
Purpose Get the current TTS language.

REQUEST

PARAMETERS

NAME	TYPE	MANDATORY	ADDITIONAL

RESPONSE

PARAMETERS

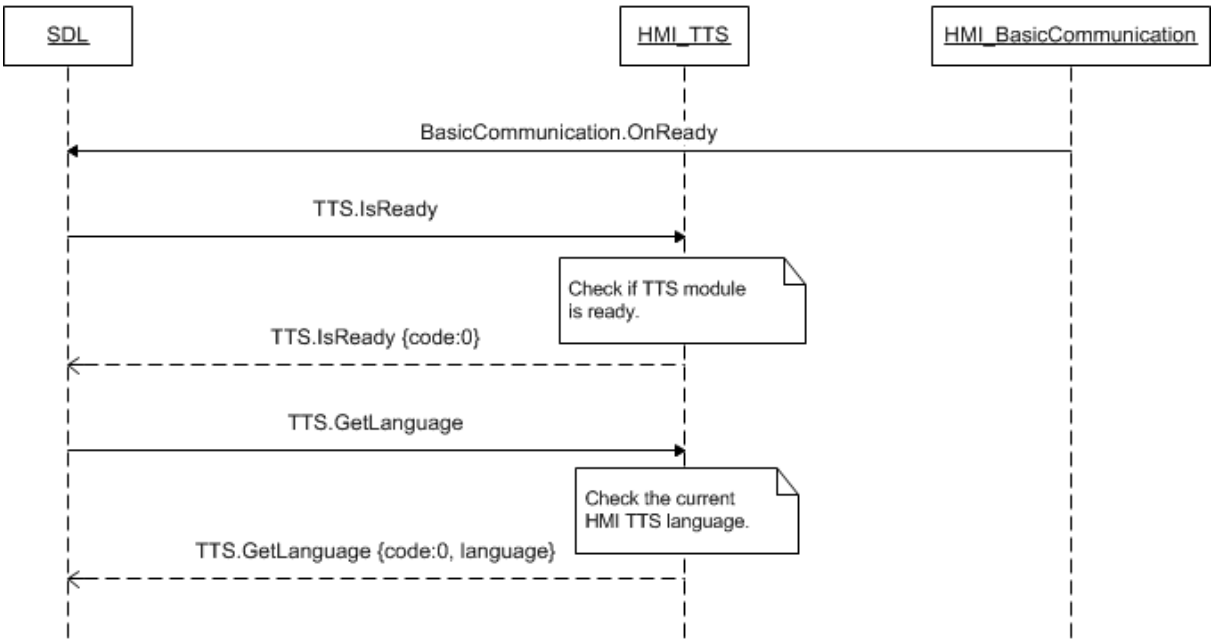
NAME	TYPE	MANDATORY	ADDITIONAL
language	Common.Language	true	

Sequence Diagrams

SEQUENCE DIAGRAM

GetLanguage

View Diagram



Example Request

```
{
 "id" : 110,
 "jsonrpc" : "2.0",
 "method" : "TTS.GetLanguage",
}
```

Example Response

```
{
 "id" : 110,
 "jsonrpc" : "2.0",
 "result" :
 {
  "language" : "DE-DE",
  "code" : 0,
  "method" : "TTS.GetLanguage"
 }
}
```

Example Error


```
{
 "id" : 110,
 "jsonrpc" : "2.0",
 "error" :
 {
  "code" : 22,
  "message" : "During the API call the unknown error has occurred",
  "data" :
  {
   "method" : "TTS.GetLanguage"
  }
 }
}
```


GetSupportedLanguages

Type Function
Sender SDL
Purpose Get the supported TTS languages

REQUEST

PARAMETERS

NAME	TYPE	MANDATORY	ADDITIONAL

RESPONSE

PARAMETERS

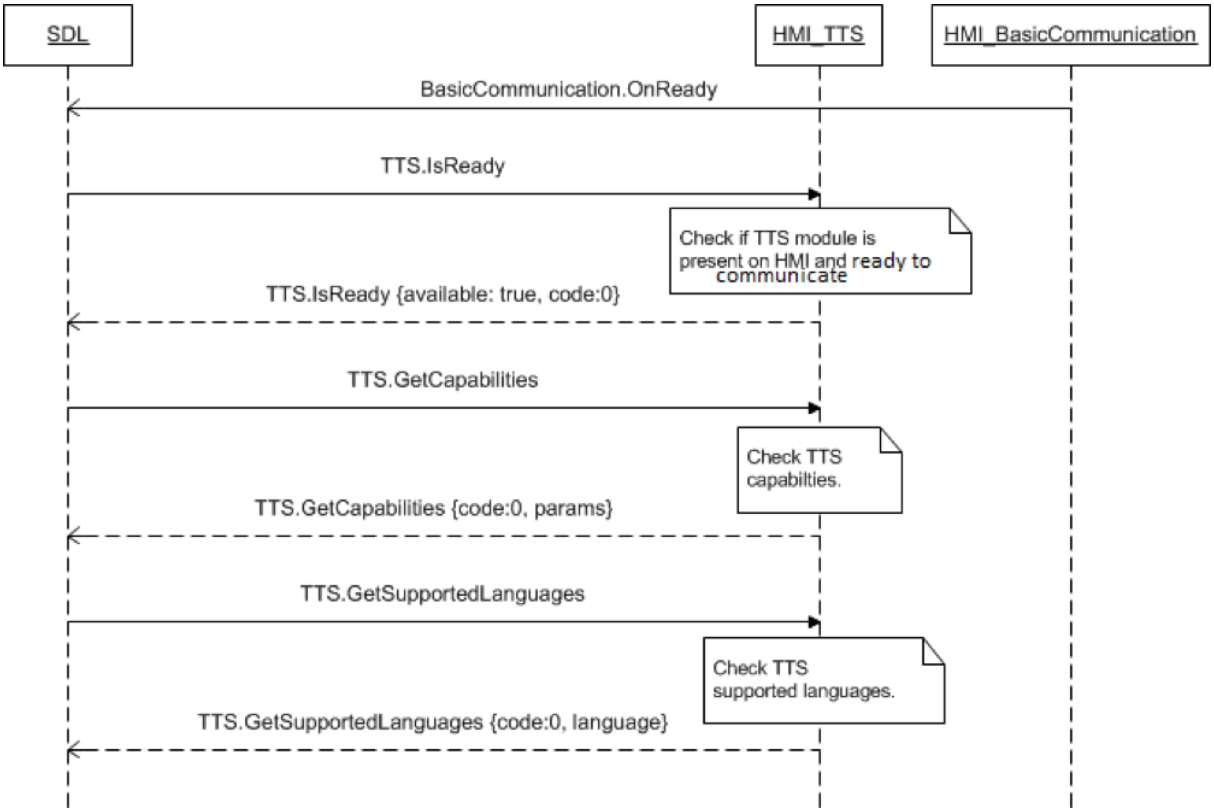
NAME	TYPE	MANDATORY	ADDITIONAL
languages	Common.Language	true	array: true minsize: 1 maxsize: 100

Sequence Diagrams

SEQUENCE DIAGRAM

GetSupportedLanguages

View Diagram



Example Request

```
{
 "id" : 19,
 "jsonrpc" : "2.0",
 "method" : "TTS.GetSupportedLanguages"
}
```

Example Response


```
{
 "id" : 19,
 "jsonrpc" : "2.0",
 "result" :
 {
  "languages" : [AR-SA, DE-DE, EN-GB, EN-US, ES-ES, FR-FR, IT-IT],
  "code" : 0,
  "method" : "TTS.GetSupportedLanguages"
 }
}
```


Example Error


```
{
 "id" : 19,
 "jsonrpc" : "2.0",
 "error" :
 {
  "code" : 11,
  "message" : "The data sent is invalid",
  "data" :
  {
   "method" : "TTS.GetSupportedLanguages"
  }
 }
}
```

IsReady

Type

Function

Sender

SDL

Purpose

Request ready state of TTS Module

REQUEST

NOTE

1. SDL sends `TTS.IsReady` request after HMI confirms its readiness via `BC.OnReady` notification.
2. If HMI responds with `"available":false`, SDL will not further communicate over TTS interface with HMI.
3. If HMI does not respond during SDL's default timeout, SDL will continue to send RPCs over TTS interface to HMI.

PARAMETERS

The request does not have specific parameters.

RESPONSE

MUST

1. Check whether TTS component is available and ready.
2. Respond correspondingly to results of this check.

PARAMETERS

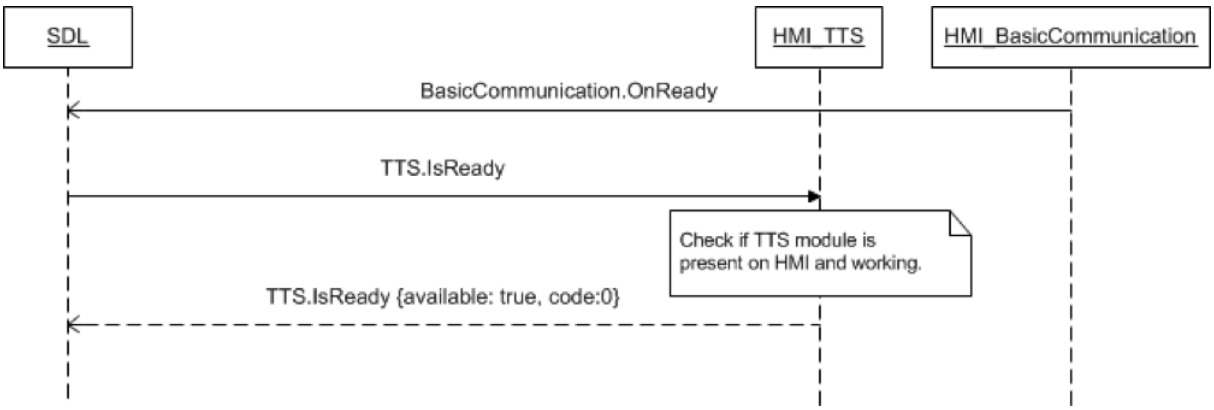
NAME	TYPE	MANDATORY	ADDITIONAL
available	Boolean	true	

Sequence Diagrams

SEQUENCE DIAGRAM

IsReady

View Diagram



Example Request

```
{
 "id" : 45,
 "jsonrpc" : "2.0",
 "method" : "TTS.IsReady"
}
```

Example Response

```
{
 "id" : 45,
 "jsonrpc" : "2.0",
 "result" :
 {
  "available" : true,
  "code" : 0,
  "method" : "TTS.IsReady"
 }
}
```

Example Error

```
{
 "id" : 45,
 "jsonrpc" : "2.0",
 "error" :
 {
  "code" : 11,
  "message" : "Invalid data",
  "data" :
  {
   "method" : "TTS.IsReady"
  }
 }
}
```

OnLanguageChange

Type Notification
Sender HMI
Purpose Inform SDL that the language for the TTS component has changed.

Notification

PARAMETERS

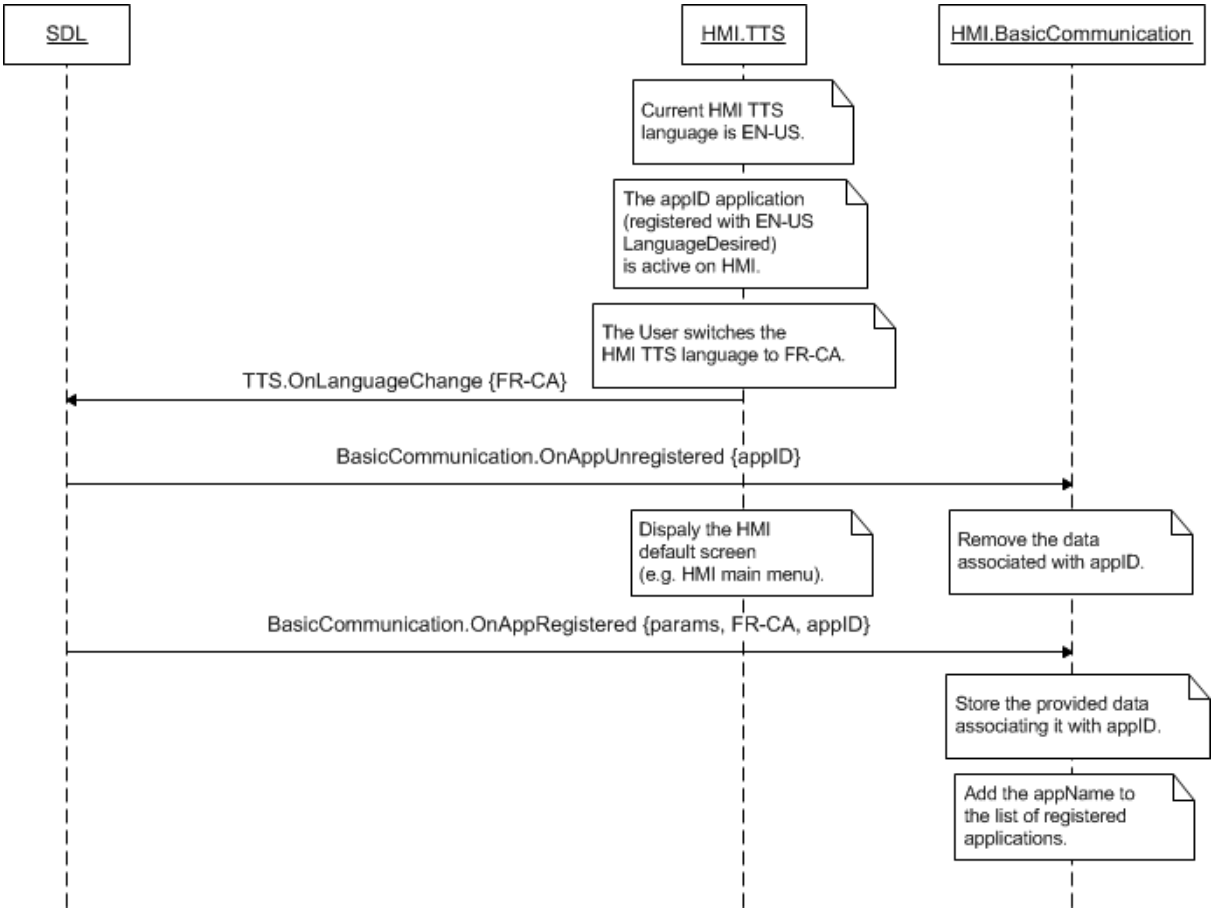
NAME	TYPE	MANDATORY	ADDITIONAL
language	Common.Language	true	

Sequence Diagrams

SEQUENCE DIAGRAM

OnLanguageChange

View Diagram



JSON EXAMPLE NOTIFICATION

```
{
 "jsonrpc" : "2.0",
 "method" : "TTS.OnLanguageChange",
 "params" :
 {
  "language" : "IT-IT"
 }
}
```

OnResetTimeout

Type

Notification

Sender

HMI

Purpose

Inform SDL to reset the timeout if speaking is running over the default timeout.

Notification

PARAMETERS

NAME	TYPE	MANDATORY	ADDITIONAL
appID	Integer	true	
methodName	String	true	

Sequence Diagrams

SEQUENCE DIAGRAM

OnResetTimeout for Speak SUCCESS

View Diagram

SEQUENCE DIAGRAM

OnResetTimeout for Speak GENERIC_ERROR

View Diagram

JSON EXAMPLE NOTIFICATION

```
{
 "jsonrpc" : "2.0",
 "method" : "TTS.OnResetTimeout",
    "params" :
 {
  "appID" : 123,
  "methodName" : "Speak"

 }
}
```

SetGlobalProperties

Type Function
Sender SDL
Purpose Set the properties for the TTS component.

Description

SDL requests to set the values for the prompts to be spoken by TTS during the User's interaction with the application over head unit.

REQUEST

On receiving `AddCommand` with `CommandType = Command` before a custom `helpPrompt` is set by the application, SDL must send updated values of `helpPrompt` via TTS.SetGlobalProperties request to HMI.

PARAMETERS

NAME	TYPE	MANDATORY	ADDITIONAL
helpPrompt	Common.TTSChunk	false	array: true minsize: 0 maxsize: 100
timeoutPrompt	Common.TTSChunk	false	array: true minsize: 1 maxsize: 100
appID	Integer	true	

RESPONSE

PARAMETERS

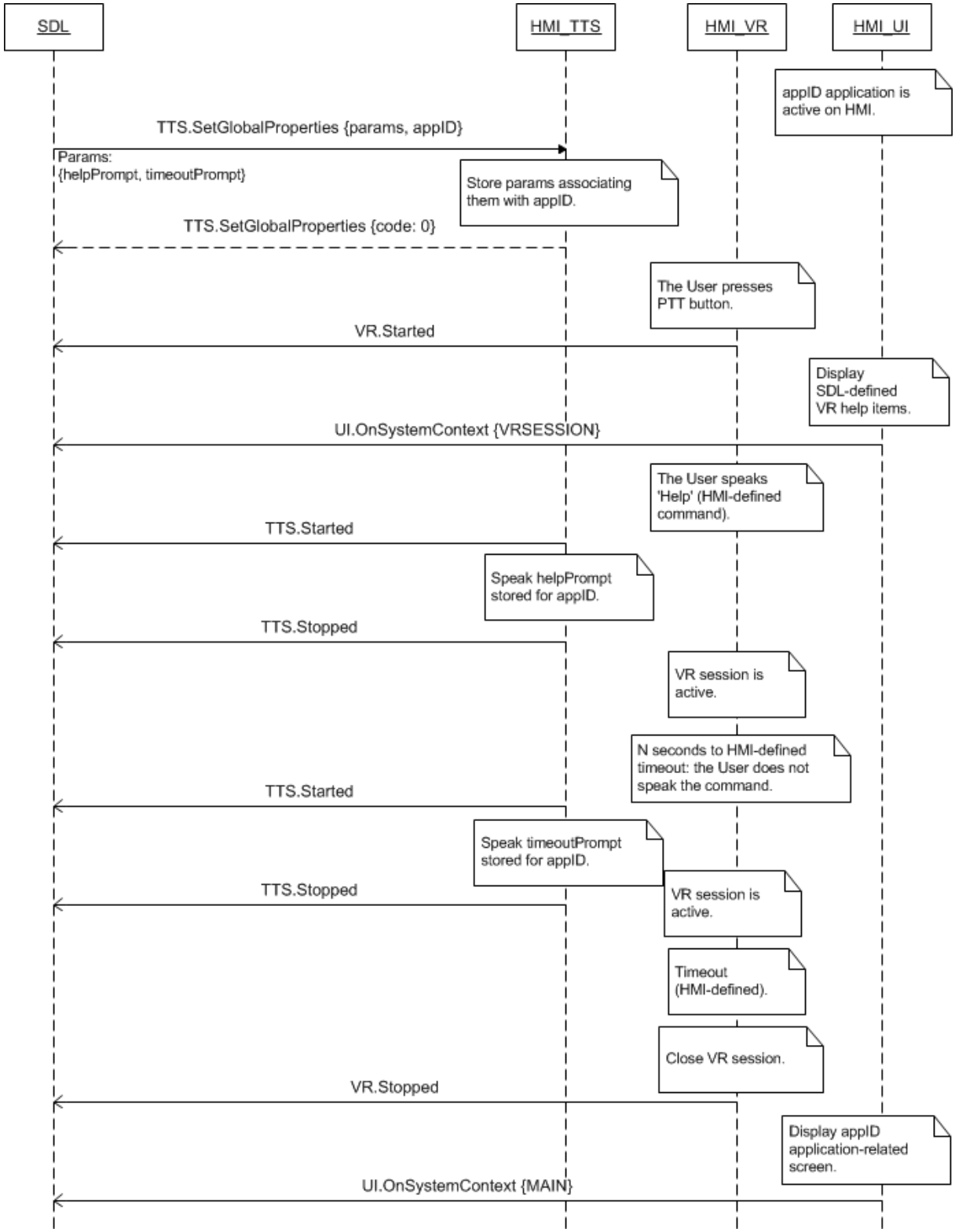
This RPC has no additional parameter requirements

Sequence Diagrams

SEQUENCE DIAGRAM

SetGlobalProperties

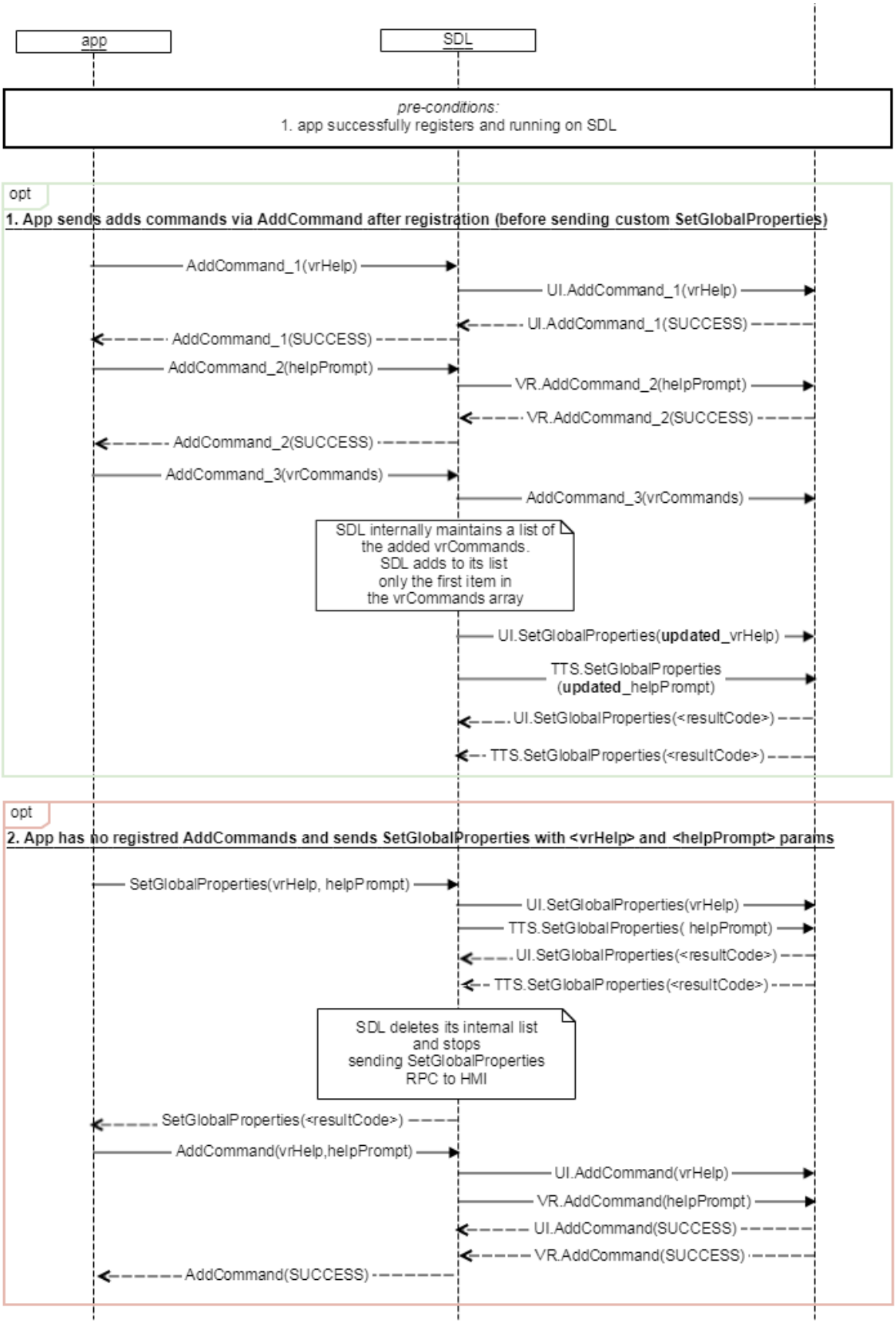
View Diagram



SEQUENCE DIAGRAM

SetGlobalProperties request with VRHelp and HelpPrompt params

View Diagram



Example Request

```
{
 "id" : 37,
 "jsonrpc" : "2.0",
 "method" : "TTS.SetGlobalProperties",
 "params" :
 {
  "helpPrompt" :
  [
   {
    "text" : "Yes",
    "type" : "TEXT"
   },
   {
    "text" : "No",
    "type" : "TEXT"
   },
   {
    "text" : "Skip",
    "type" : "TEXT"
   }
  ],

  "timeoutPrompt" :
  [
   {
    "text" : "Please make a choice",
    "type" : "TEXT"
   },
   {
    "text" : "The time is about to expire",
    "type" : "TEXT"
   }
  ],
  "appID" : 65542
 }
}
```

Example Response

```
{
 "id" : 37,
 "jsonrpc" : "2.0",
 "result" :
 {
  "code" : 0,
  "method" : "TTS.SetGlobalProperties"
 }
}
```

Example Error

```
{
 "id" : 37,
 "jsonrpc" : "2.0",
 "error" :
 {
  "code" : 2,
  "message" : "TTS is not supported",
  "data" :
  {
   "method" : "TTS.SetGlobalProperties"
  }
 }
}
```

Speak

Type

Function

Sender

SDL

Purpose

Prompt TTS to speak the requested text to the user.

REQUEST

PARAMETERS

NAME	TYPE	MANDATORY	ADDITIONAL
ttsChunks	Common.TTSChunk	true	array: true minsize: 1 maxsize: 100
appID	Integer	true	
speakType	Common.MethodName	false	
playTone	Boolean	false	

RESPONSE

PARAMETERS

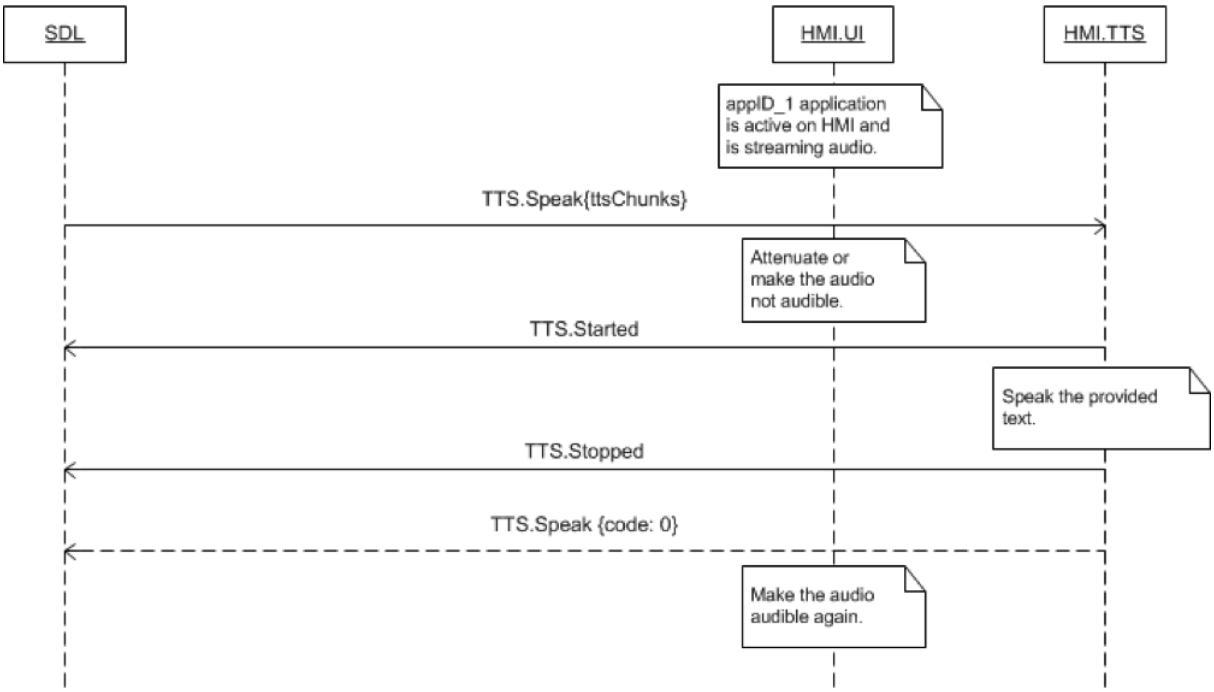
NAME	TYPE	MANDATORY	ADDITIONAL

Sequence Diagrams

SEQUENCE DIAGRAM

Speak

View Diagram



Example Request

```
{
 "id" : 144,
 "jsonrpc" : "2.0",
 "method" : "TTS.Speak",
 "params" :
 {
  "ttsChunks" :
  [
   {
    "text" : "Please say a command",
    "type" : "TEXT"
   }
  ]
 }
}
```

Example Response


```
{
 "id" : 144,
 "jsonrpc" : "2.0",
 "result" :
 {
  "code" : 0,
  "method" : "TTS.Speak"
 }
}
```


Example Error


```
{
 "id" : 144,
 "jsonrpc" : "2.0",
 "error" :
 {
  "code" : 5,
  "message" : "The command was aborted",
  "data" :
  {
   "method" : "TTS.Speak"
  }
 }
}
```

Started

Type Notification
Sender HMI
Purpose Inform SDL that TTS has started.

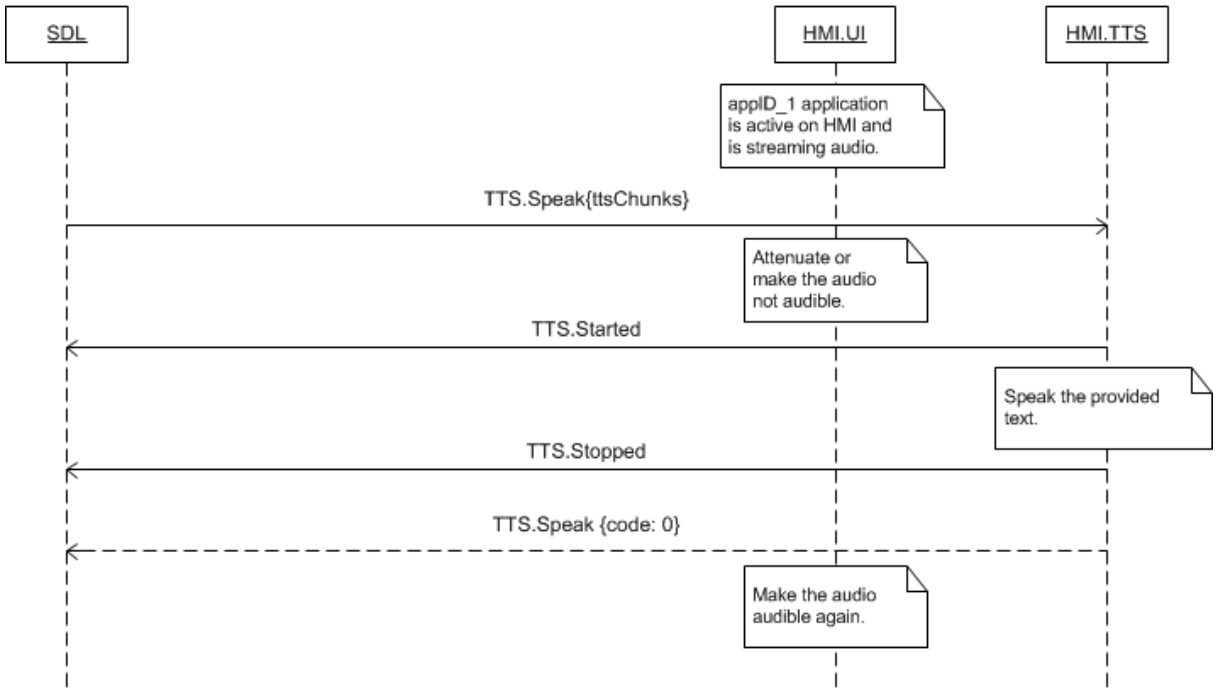
Notification

Sequence Diagrams

SEQUENCE DIAGRAM

Started upon TTS.Speak request from SDL

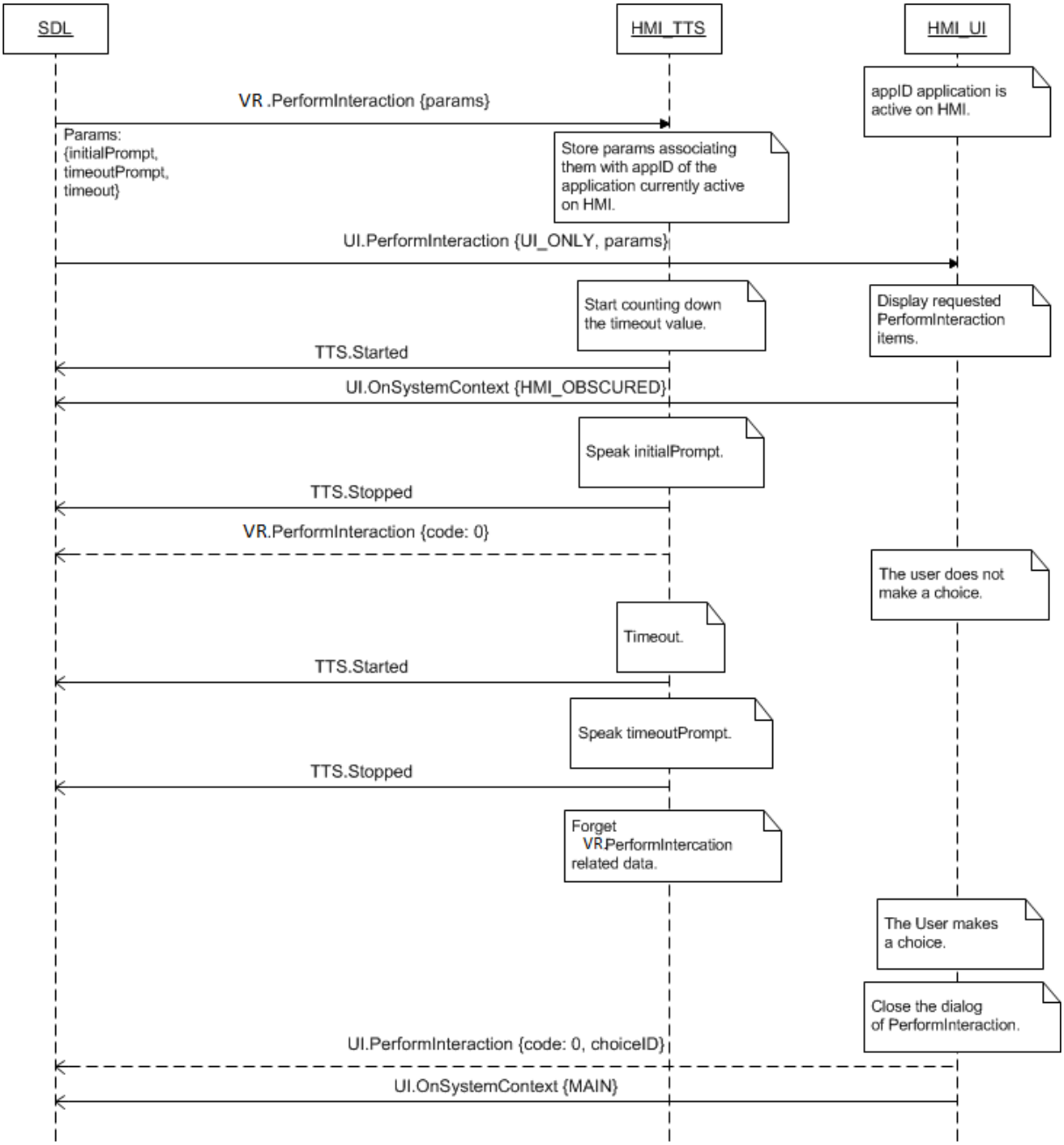
View Diagram



SEQUENCE DIAGRAM

Started during PerformInteraction

View Diagram



Example Request

```
{
  "jsonrpc" : "2.0",
  "method" : "TTS.Started",
}
```

Example Response


```
{
 "id" : 37,
 "jsonrpc" : "2.0",
 "result" :
 {
  "code" : 0,
  "method" : "TTS.Started"
 }
}
```


Example Error


```
{
 "id" : 37,
 "jsonrpc" : "2.0",
 "error" :
 {
  "code" : 22,
  "message" : "Something went wrong",
  "data" :
  {
   "method" : "TTS.Started"
  }
 }
}
```

Stopped

Type Notification
Sender HMI
Purpose Inform SDL that TTS has stopped.

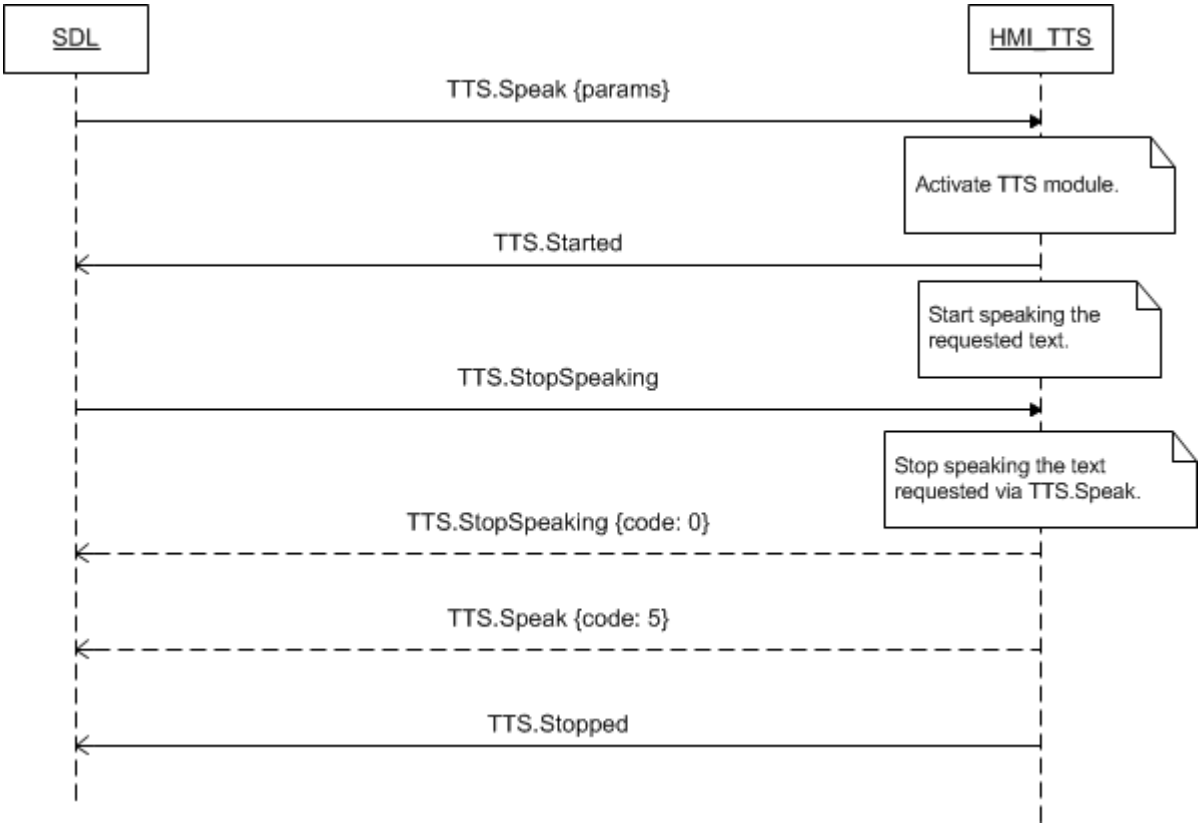
Notification

Sequence Diagrams

SEQUENCE DIAGRAM

Stopped after StopSpeaking from SDL

View Diagram



Example Request

```
{
  "jsonrpc" : "2.0",
  "method" : " TTS.Stopped",
}
```

Example Response


```
{
 "id" : 37,
 "jsonrpc" : "2.0",
 "result" :
 {
  "code" : 0,
  "method" : "TTS.Stopped"
 }
}
```


Example Error


```
{
 "id" : 37,
 "jsonrpc" : "2.0",
 "error" :
 {
  "code" : 22,
  "message" : "Something went wrong",
  "data" :
  {
   "method" : "TTS.Stopped"
  }
 }
}
```

StopSpeaking

Type Function
Sender SDL
Purpose Interrupt the current spoken session.

REQUEST

PARAMETERS

NAME	TYPE	MANDATORY	ADDITIONAL

RESPONSE

PARAMETERS

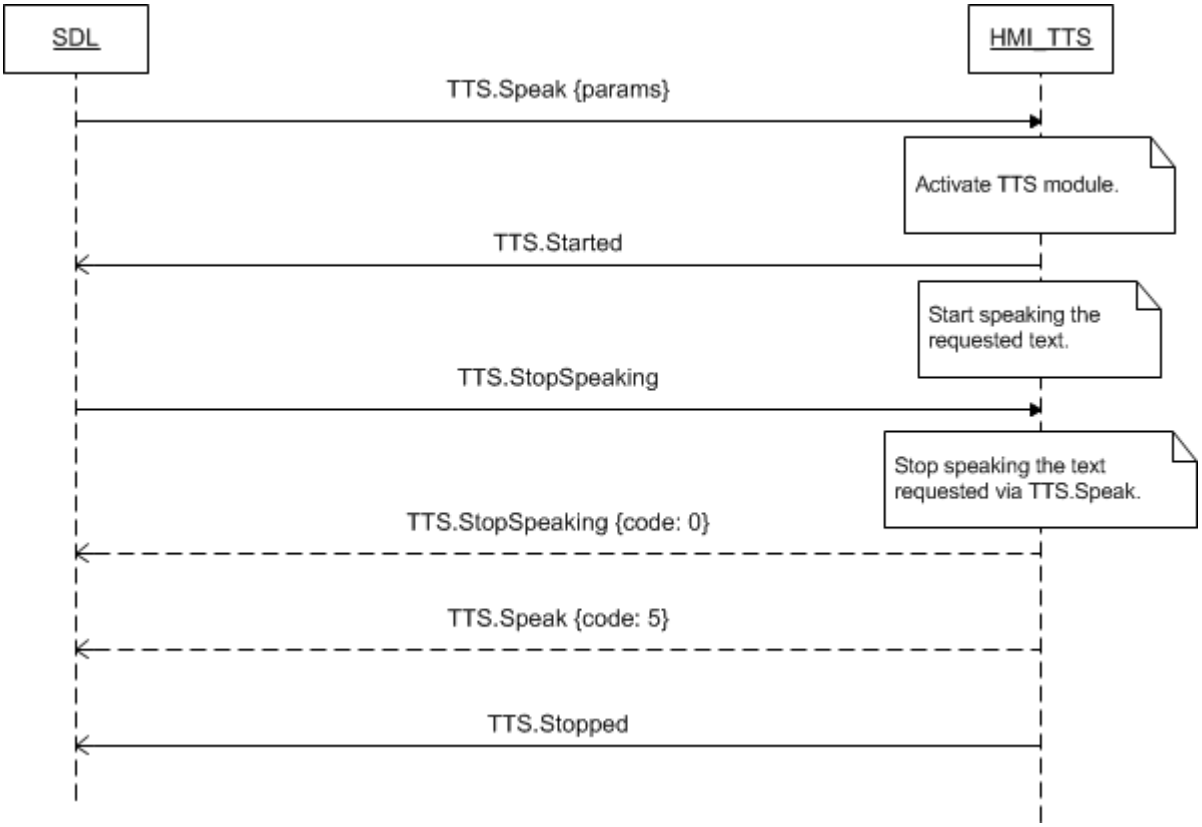
This RPC has no additional parameter requirements

Sequence Diagrams

SEQUENCE DIAGRAM

StopSpeaking

View Diagram



Example Request

```
{
  "id" : 148,
  "jsonrpc" : "2.0",
  "method" : "TTS.StopSpeaking"
}
```

Example Response


```
{
 "id" : 148,
 "jsonrpc" : "2.0",
 "result" :
 {
  "code" : 0,
  "method" : "TTS.StopSpeaking"
 }
}
```


Example Error


```
{
 "id" : 148,
 "jsonrpc" : "2.0",
 "error" :
 {
  "code" : 11,
  "message" : "Invalid data: invalid JSON syntax",
  "data" :
  {
   "method" : "TTS.StopSpeaking"
  }
 }
}
```

AddCommand

Type Function
Sender SDL
Purpose Add a command to the specified application's menu or sub menu

UI.AddCommand represents a request from an application to add a command to the application's menu or sub-menu. This RPC can be sent to the HMI for an application that is registered and in any state (FULL, BACKGROUND, etc.)

MUST

1. The commands sent for the application via AddCommand must be accessible from a Menu
2. The user must be able to enter the Menu while the related application is in the FULL state
3. Store the data provided in this RPC with the requesting application's appID
4. Add the command to the application's menu at the position specified in the `menuParams`

NOTE

- If SDL sends the HMI a UI.AddCommand and a VR.AddCommand, and receives a SUCCESS from one and a failure from the other, SDL will send a UI.DeleteCommand for the AddCommand which succeeded.
- If the `menuParams` contains a `parentID` the command is part of a sub menu. SDL adds SubMenu Commands to the top level Menu via UI.AddSubMenu

REQUEST

PARAMETERS

NAME	TYPE	MANDATORY	ADDITIONAL
cmdID	Integer	true	minvalue: 0 maxvalue: 2000000000
menuParams	Common.MenuParams	false	
cmdIcon	Common.Image	false	
appID	Integer	true	

RESPONSE

PARAMETERS

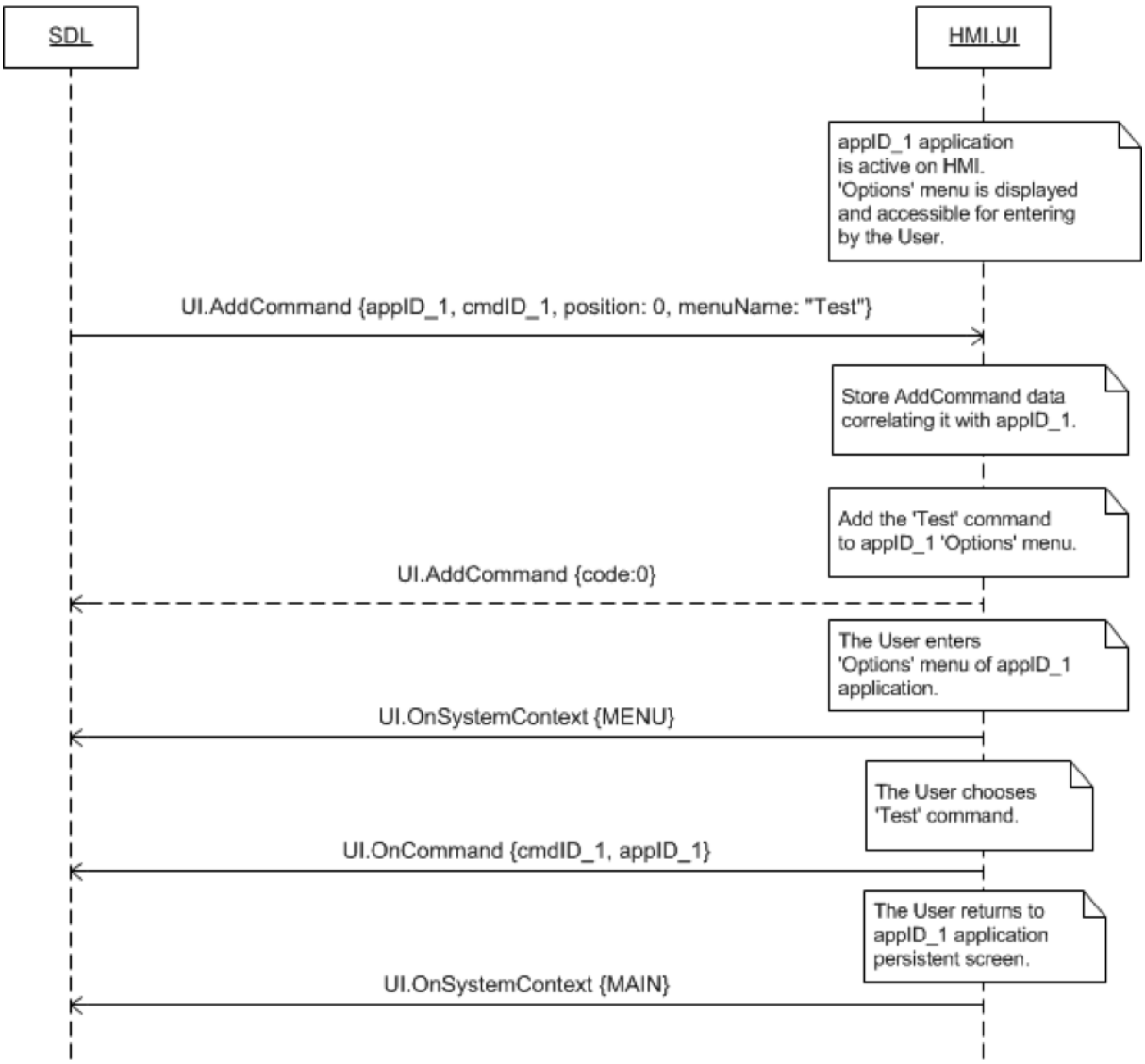
This RPC has no additional parameter requirements

Sequence Diagrams

SEQUENCE DIAGRAM

AddCommand Command Chosen By User

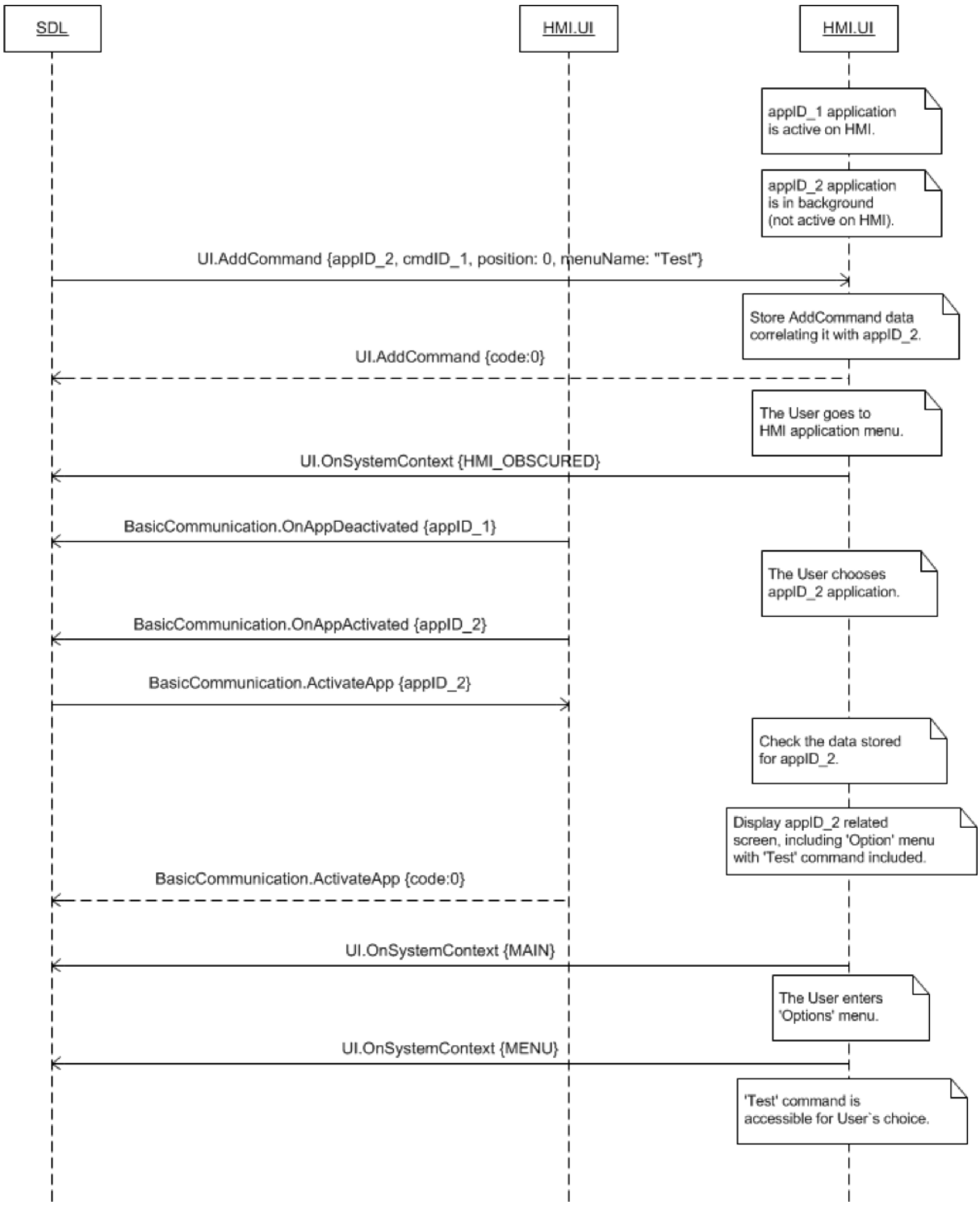
View Diagram



SEQUENCE DIAGRAM

AddCommand App Inactive

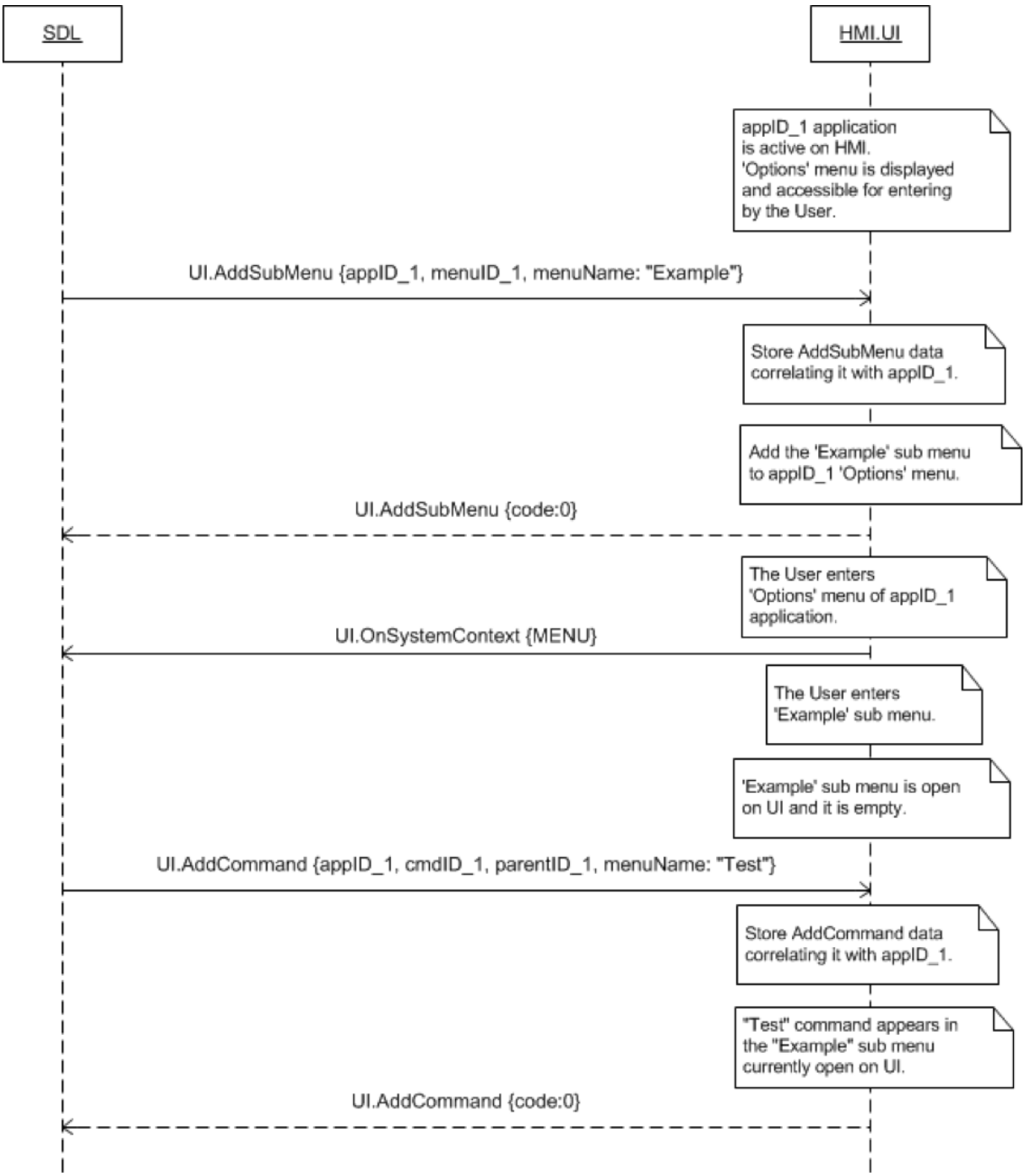
View Diagram



SEQUENCE DIAGRAM

AddCommand with sub menu

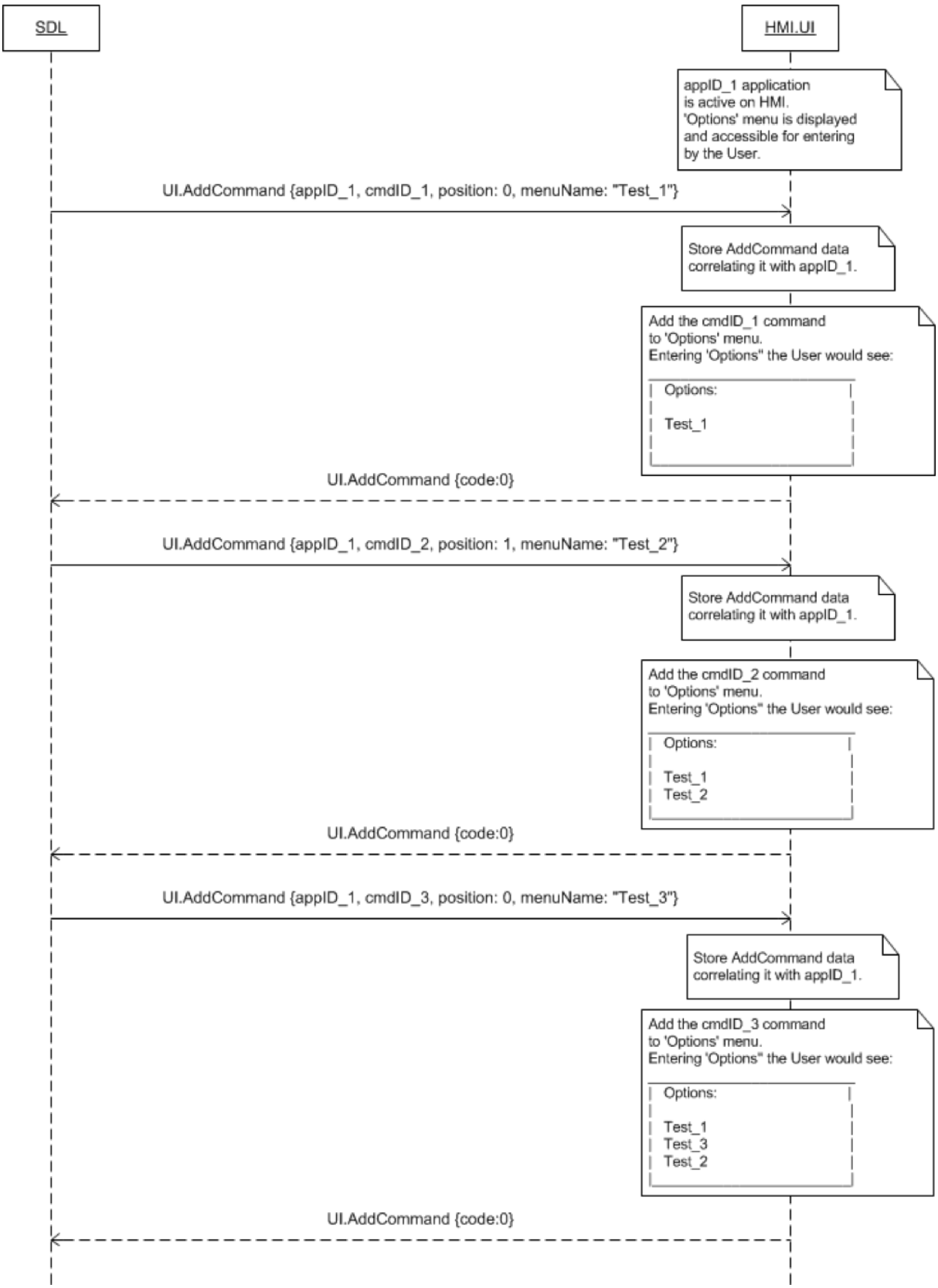
View Diagram



SEQUENCE DIAGRAM

AddCommand positions

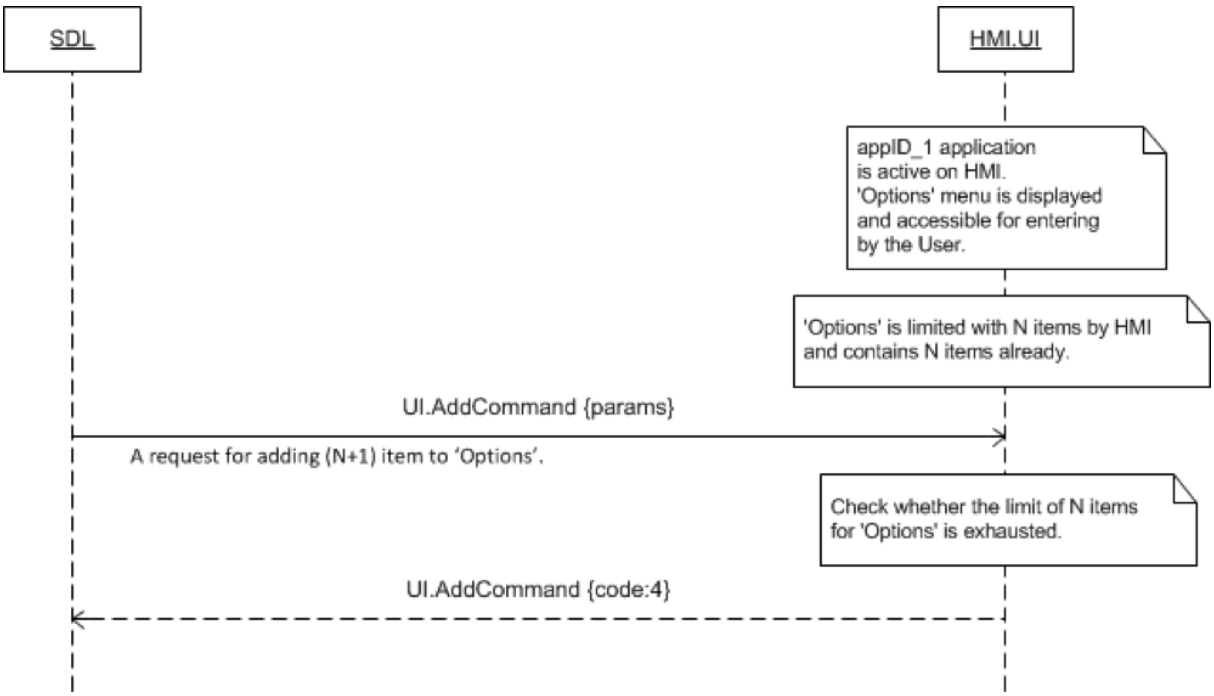
View Diagram



SEQUENCE DIAGRAM

AddCommand Rejected Limit Reached

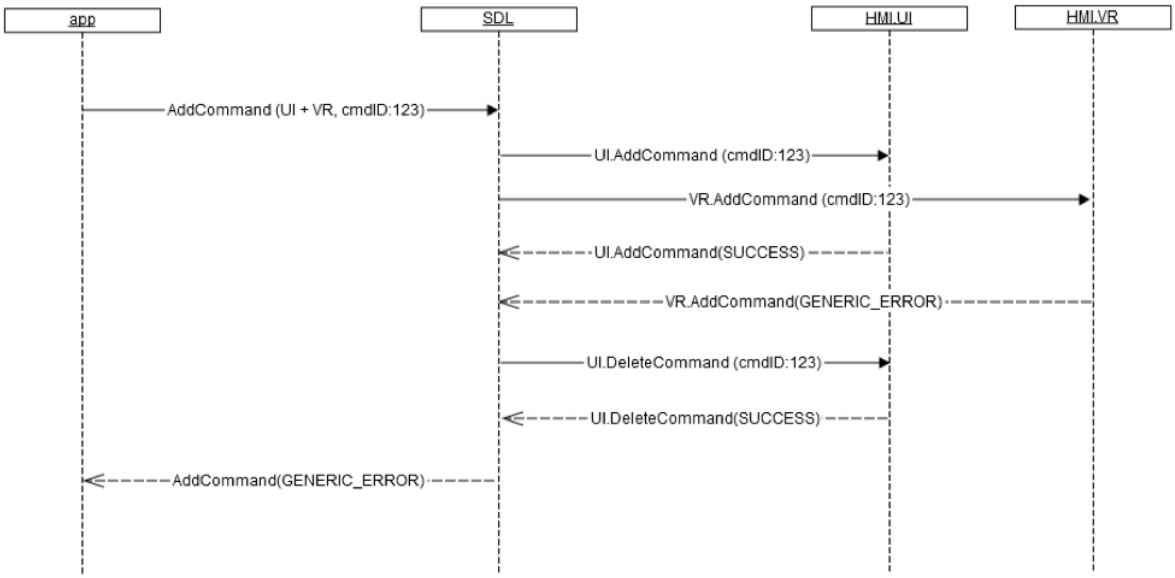
View Diagram



SEQUENCE DIAGRAM

AddCommand UI Succeeds, VR Fails

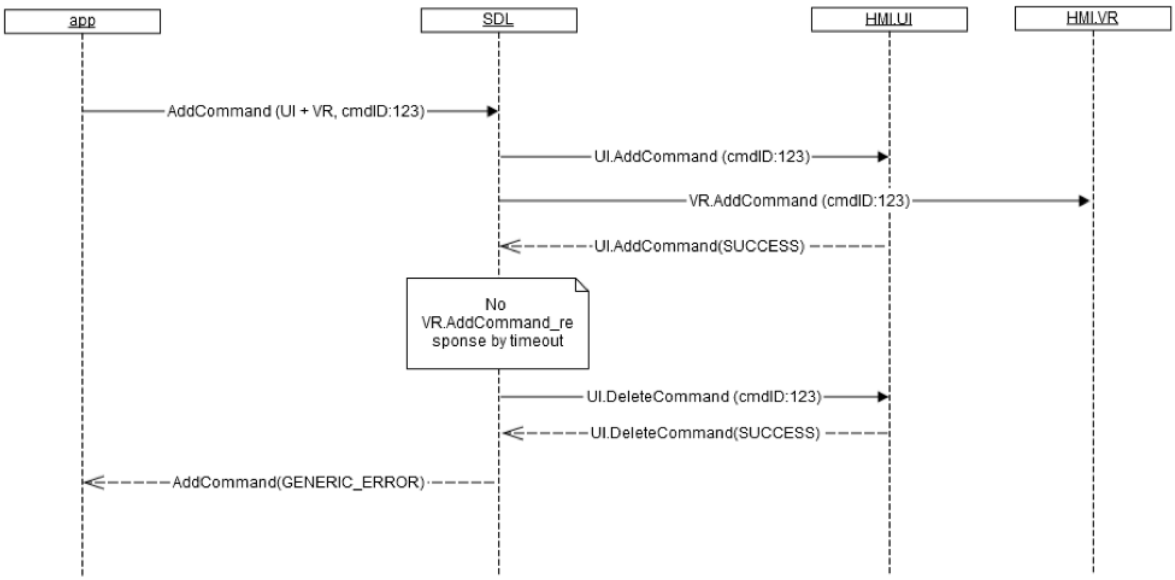
View Diagram



SEQUENCE DIAGRAM

AddCommand UI Succeeds, VR Unresponsive

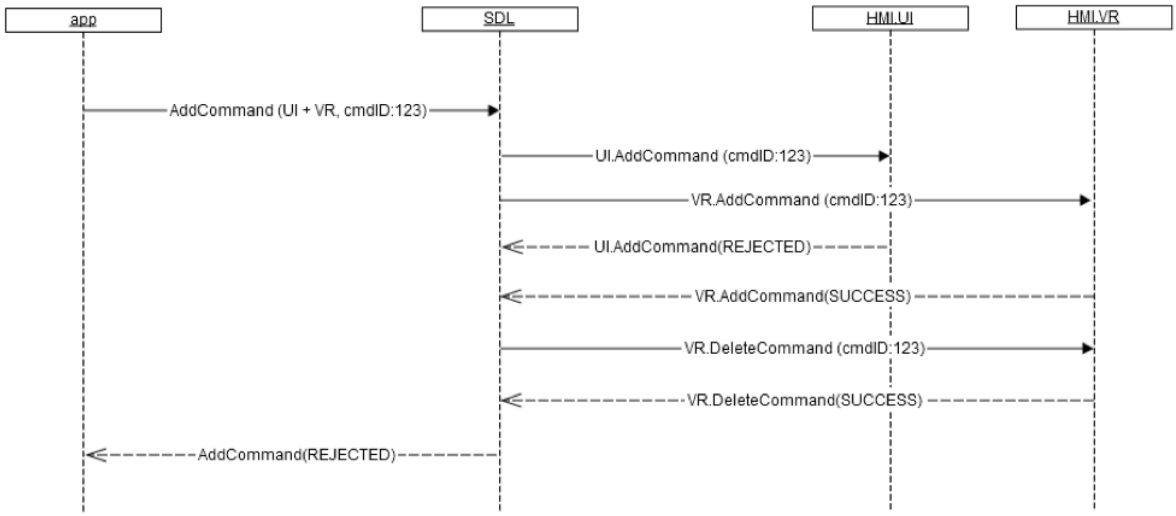
View Diagram



SEQUENCE DIAGRAM

AddCommand UI Fails, VR Succeeds

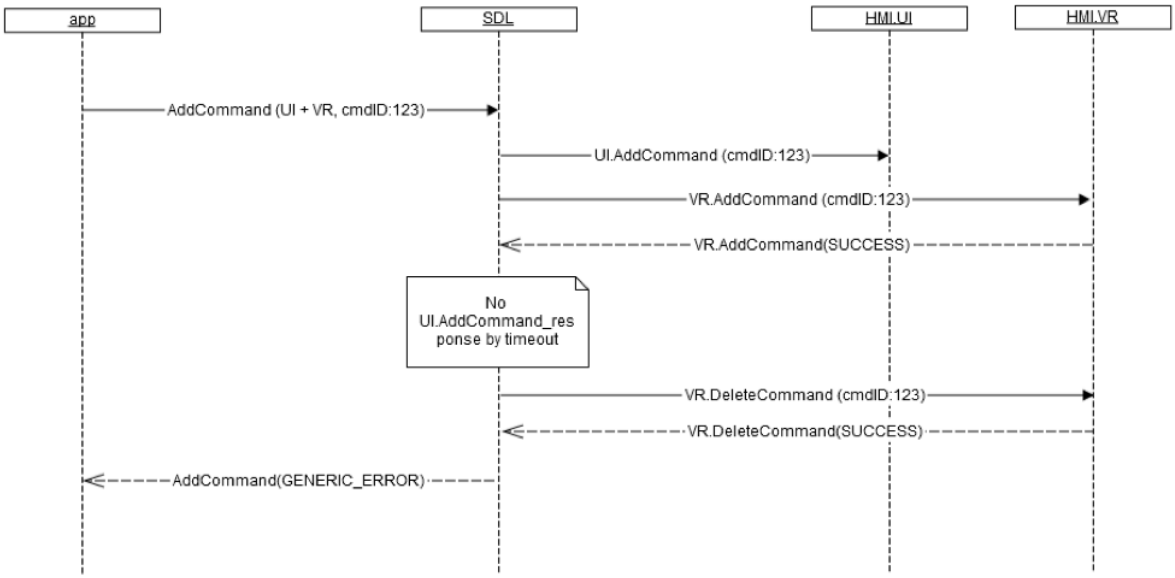
View Diagram



SEQUENCE DIAGRAM

AddCommand UI No Response, VR Succeeds

View Diagram



Example Request

```
{
 "id" : 215,
 "jsonrpc" : "2.0",
 "method" : "UI. AddCommand",
 "params" :
 {
  "cmdID" : 2318,
  "menuParams" :
   {
      "parentID" : 6,
      "position" : 0,
      "menuName" : "Show weather for tomorrow"
   },
  "cmdIcon" :
   {
      "value" : "tmp/SDL/app/Gis_meteo/1245_28.jpeg",
      "imageType" : DYNAMIC
   },
  "appID" : 65409
 }
}
```

Example Response

```
{
 "id" : 215,
 "jsonrpc" : "2.0",
 "result" :
 {
  "code" : 0,
  "method" : "UI.AddCommand"
 }
}
```

Example Error

```
{
 "id" : 215,
 "jsonrpc" : "2.0",
 "error" :
 {
  "code" : 13,
  "message" : "There’s no app with received appID registered",
  "data" :
  {
   "method" : "UI.AddCommand"
  }
 }
}
```

AddSubMenu

Type Function
Sender SDL
Purpose Add a sub menu to the specified application's menu

UI.AddSubMenu represents a request from an application to add a sub-menu to the application's menu. This RPC can be sent to the HMI for an application that is registered and in any state (FULL, BACKGROUND, etc.)

MUST

1. The sub menu sent for the application via AddCommand must be accessible from a Menu
2. If the user selects a sub menu item from the application's menu, the HMI must display all commands added via UI.AddCommand which share a `menuParams.parentID` with the menu's `menuID`
3. Store the data provided in this RPC with the requesting application's appID
4. Persist the stored data for the duration of the ignition cycle
5. Add the command to the application's menu at the position specified in the `menuParams`
6. Display images on sub menus if provided by the application.
7. Provide a WARNINGS information to the application that the SubMenu was added but no image was displayed.

8. Scale the image to ensure it fits properly in the space allocated for the display of the image. If the image does not fit properly even after scaling, the HMI shall not display any image on the submenu.

NOTE

- SDL will never request a sub menu be added to another sub menu
- HMI does not display any image in case the application does not provide an image for display on the sub menu.
- HMI does not display any image in case the image referenced by the application for display on the submenu is invalid or is not available.
- To remove an icon already sent, the app would have to delete the submenu and add it again without the icon.Otherwise, if the submenu is not deleted, another request with the same submenu id will be rejected.

REQUEST

PARAMETERS

NAME	TYPE	MANDATORY	ADDITIONAL
menuID	Integer	true	minvalue: 1 maxvalue: 2000000000
menuParams	Common.MenuParams	true	
menuIcon	Common.Image	false	
appID	Integer	true	

RESPONSE

PARAMETERS

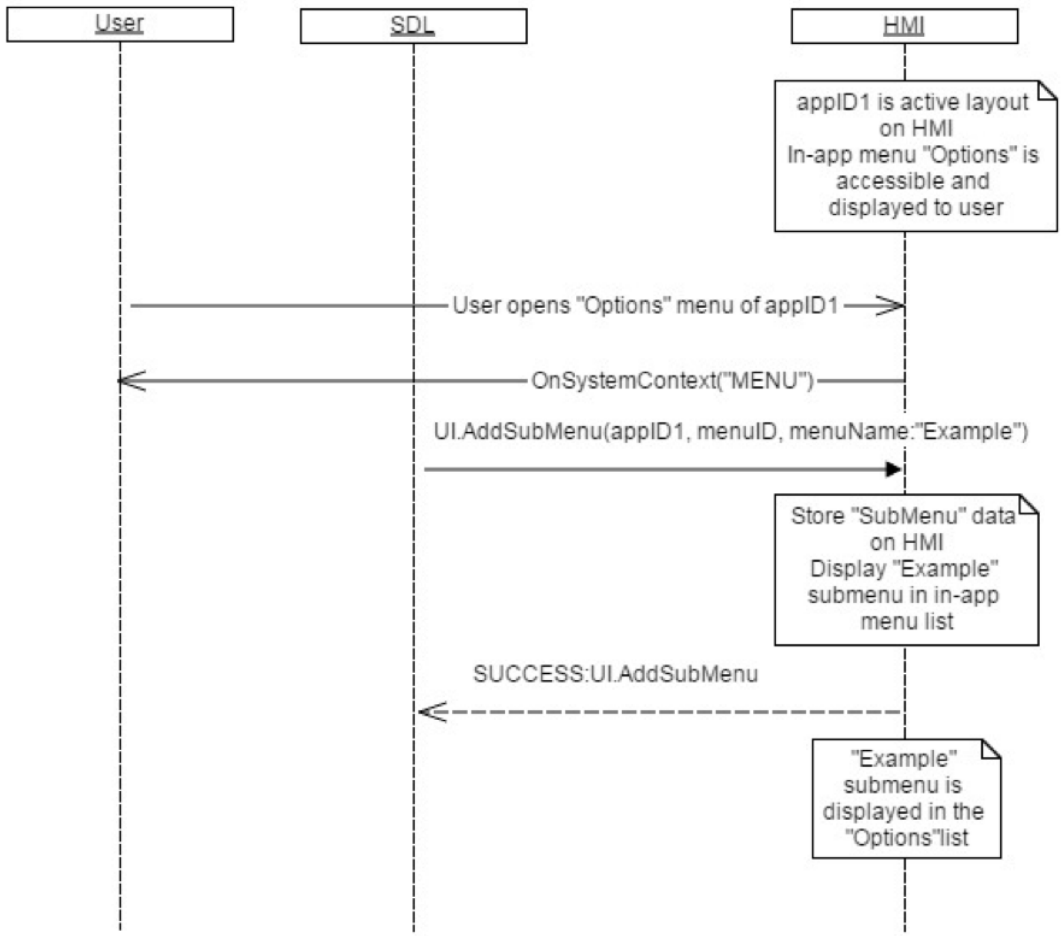
This RPC has no additional parameter requirements

Sequence Diagrams

SEQUENCE DIAGRAM

Add Sub Menu for Active App

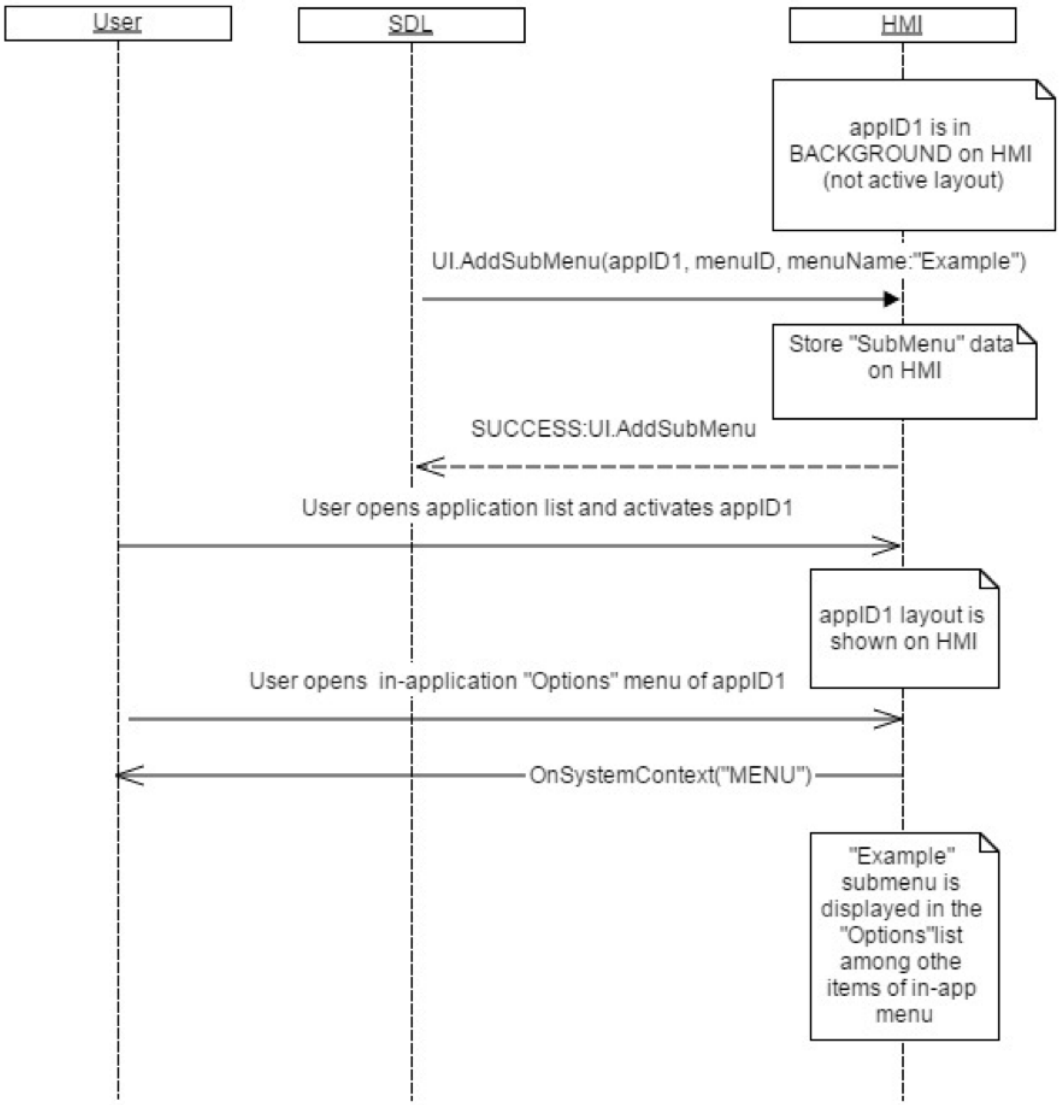
View Diagram



SEQUENCE DIAGRAM

Add Sub Menu for Inactive App

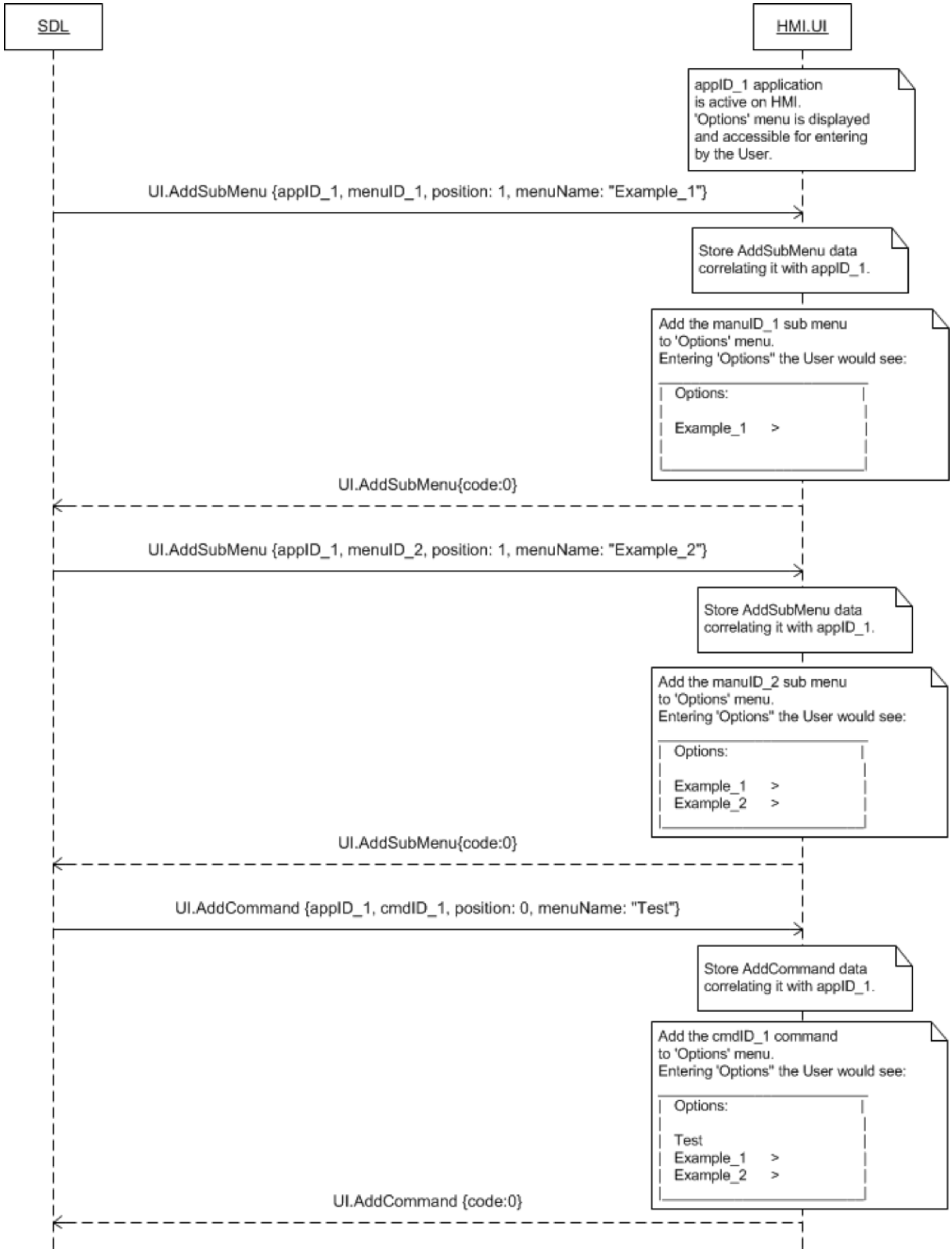
View Diagram



SEQUENCE DIAGRAM

Add Sub Menu with positions

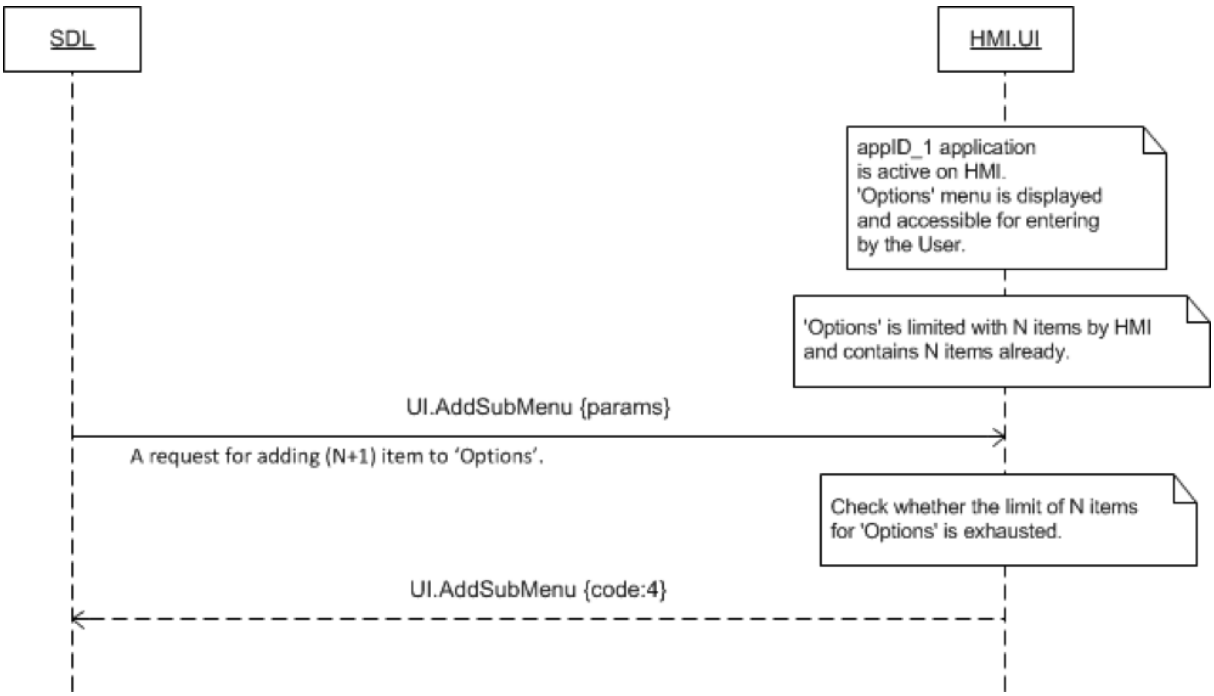
View Diagram



SEQUENCE DIAGRAM

Add Sub Menu Rejected Limit Reached

View Diagram



Example Request

```
{
 "id" : 112,
 "jsonrpc" : "2.0",
 "method" : "UI.AddSubMenu",
 "params" :
 {
  "menuID" : 345,
  "menuParams" :
  {
     "position" : 2,
     "menuName" : "Settings"
  },
  "appID" : 65464
 }
}
```

Example Response


```
{
 "id" : 112,
 "jsonrpc" : "2.0",
 "result" :
 {
  "code" : 0,
  "method" : "UI.AddSubMenu"
 }
}
```


Example Error


```
{
 "id" : 112,
 "jsonrpc" : "2.0",
 "error" :
 {
  "code" : 14,
  "message" : "Duplicate name: there was a conflict with an already
registered name of SubMenu",
  "data" :
  {
   "method" : "UI.AddSubMenu"
  }
 }
}
```

Alert

Type Function
Sender SDL
Purpose Display an alert message on the HMI

SDL sends the `UI.Alert` RPC when some information needs to be displayed to the user on a display. The Alert has a `softButtons` array of buttons which the user can use to take action on the alert.

MUST

1. If the alert includes a soft button of type `STEAL_FOCUS` and the user presses the button, the HMI must bring the app associated with the Alert into full screen mode.
2. The HMI must send UI.OnSystemContext with type `ALERT` for the application which is in `FULL` mode.
3. Respond to the Alert earlier than SDL's default timeout of 10 seconds - applicable only to alerts without `softButtons`.
4. Display the alert dialog with the text information in the `alertFields` array and optional `softButtons` and optional `displayIndicator` indicating a timeout for the alert
5. Send Buttons.OnButtonPress and/or Buttons.OnButtonEvent notifications if soft buttons associated with the alert are pressed by the user

6. Dismiss the alert after the duration has passed since receipt of the request

MAY

The HMI may provide the user with a system defined "close" button providing the user with the possibility to dismiss the alert. In this case the HMI must still respond to the alert request

NOTE

An alert may be sent to the HMI for an application which is not currently active. If the alert contains `softButtons` then the duration will be set to `0`

REQUEST

PARAMETERS

NAME	TYPE	MANDATORY	ADDITIONAL
alertStrings	Common.TextFieldStruct	true	array: true minsize: 0 maxsize: 3
duration	Integer	true	minvalue: 3000 maxvalue: 10000
softButtons	Common.SoftButton	false	array: true minsize: 0 maxsize: 4
progressIndicator	Boolean	false	
alertType	Common.AlertType	true	
appID	Integer	true	

RESPONSE

PARAMETERS

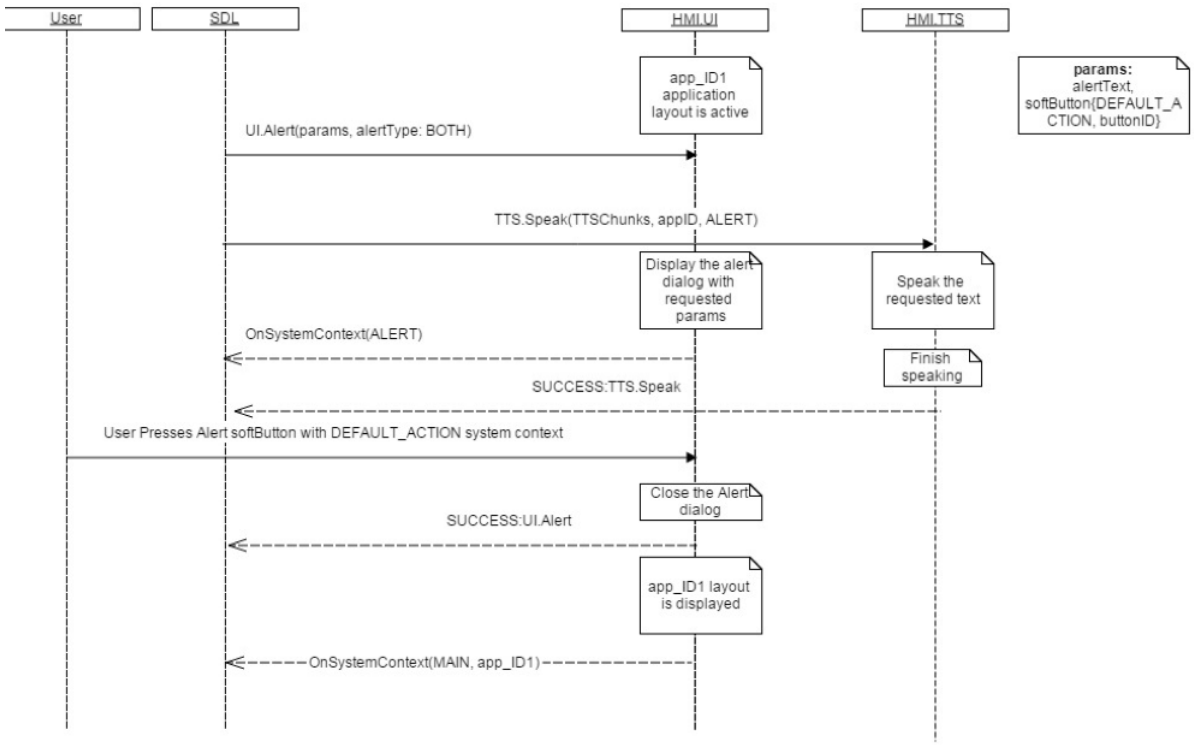
NAME	TYPE	MANDATORY	ADDITIONAL
tryAgainTime	Integer	false	minvalue: 0 maxvalue: 2000000000

Sequence Diagrams

SEQUENCE DIAGRAM

Alert closed by DEFAULT_ACTION

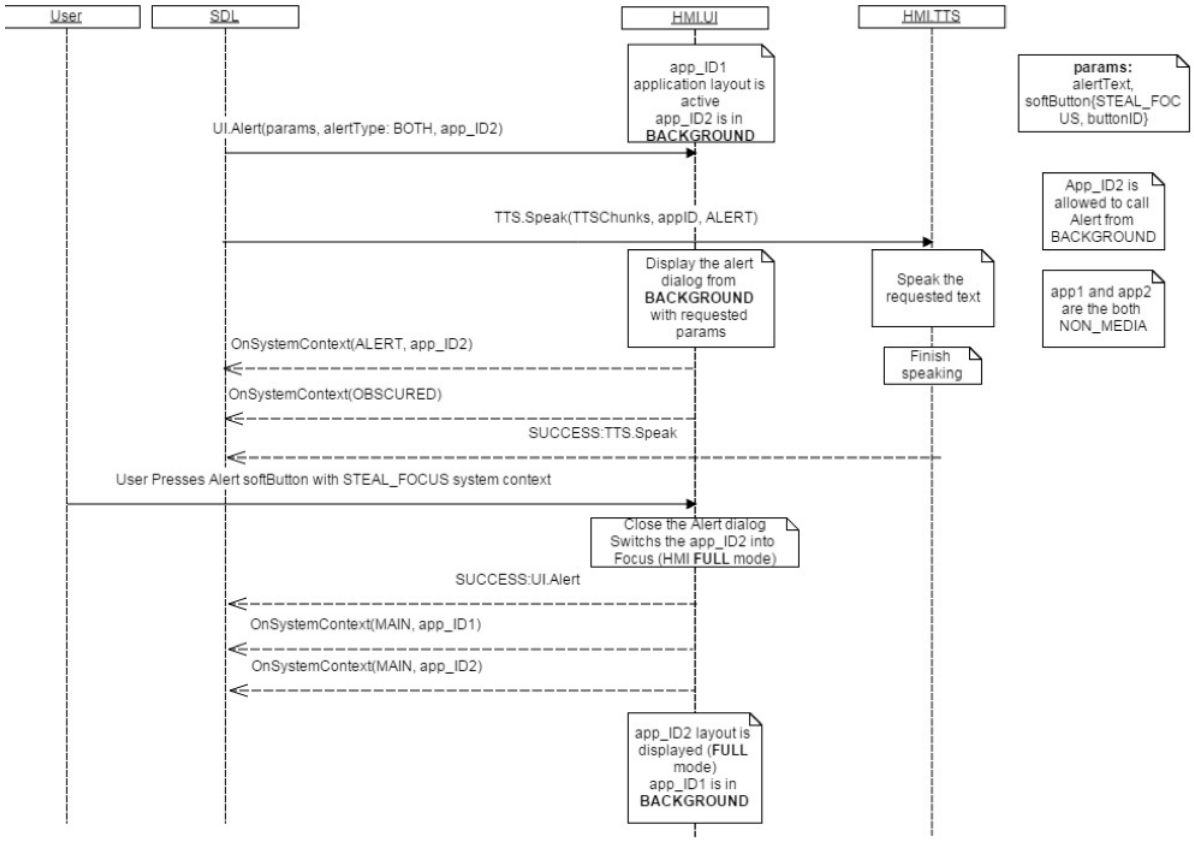
View Diagram



SEQUENCE DIAGRAM

Alert closed by STEAL_FOCUS

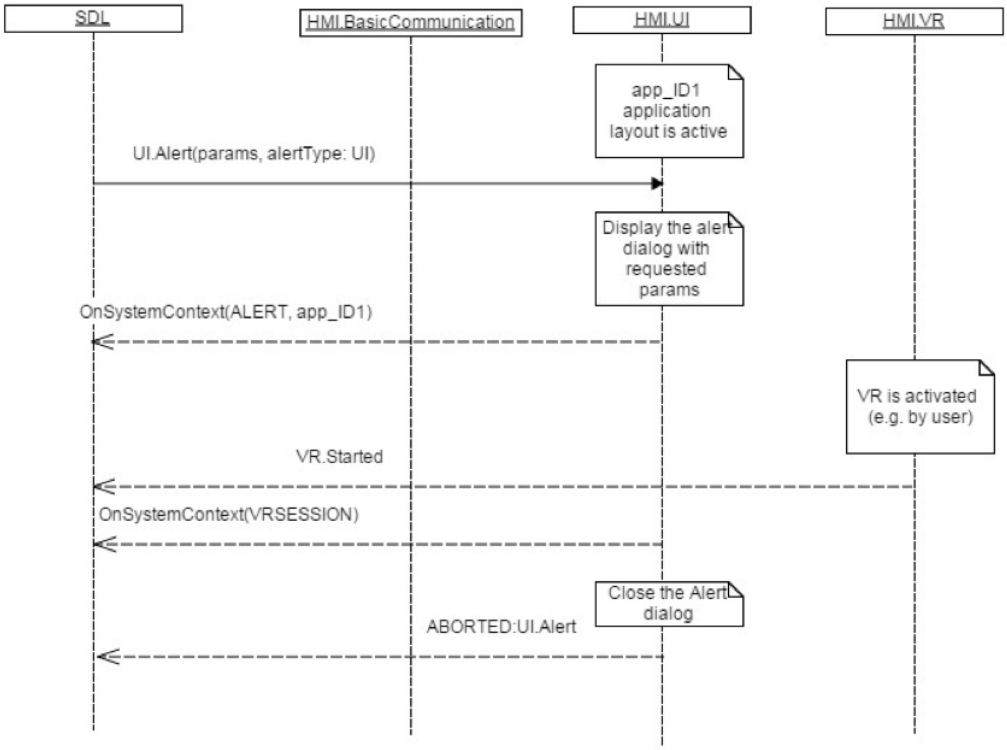
View Diagram



SEQUENCE DIAGRAM

Alert Aborted by VR Session

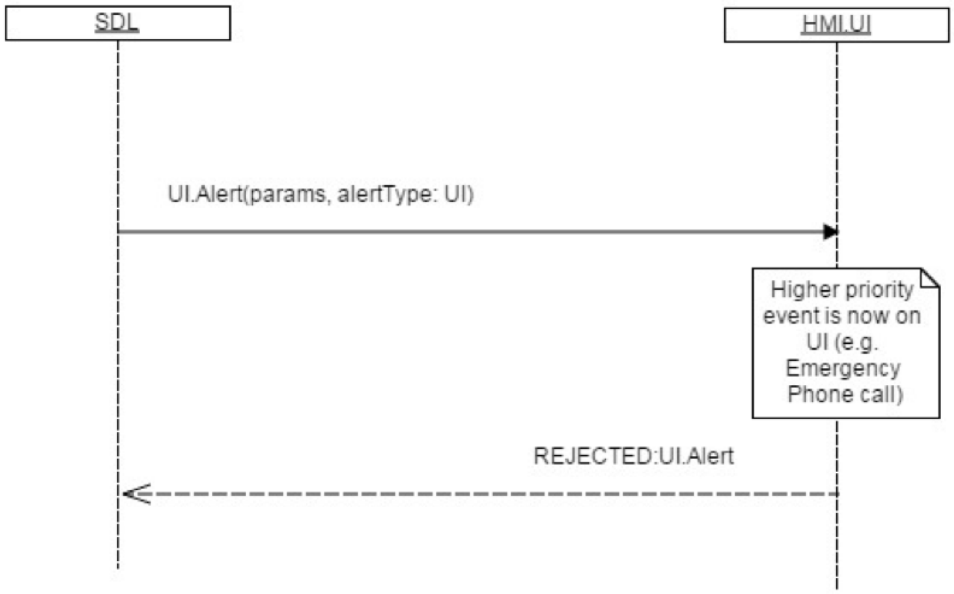
View Diagram



SEQUENCE DIAGRAM

Alert Rejected

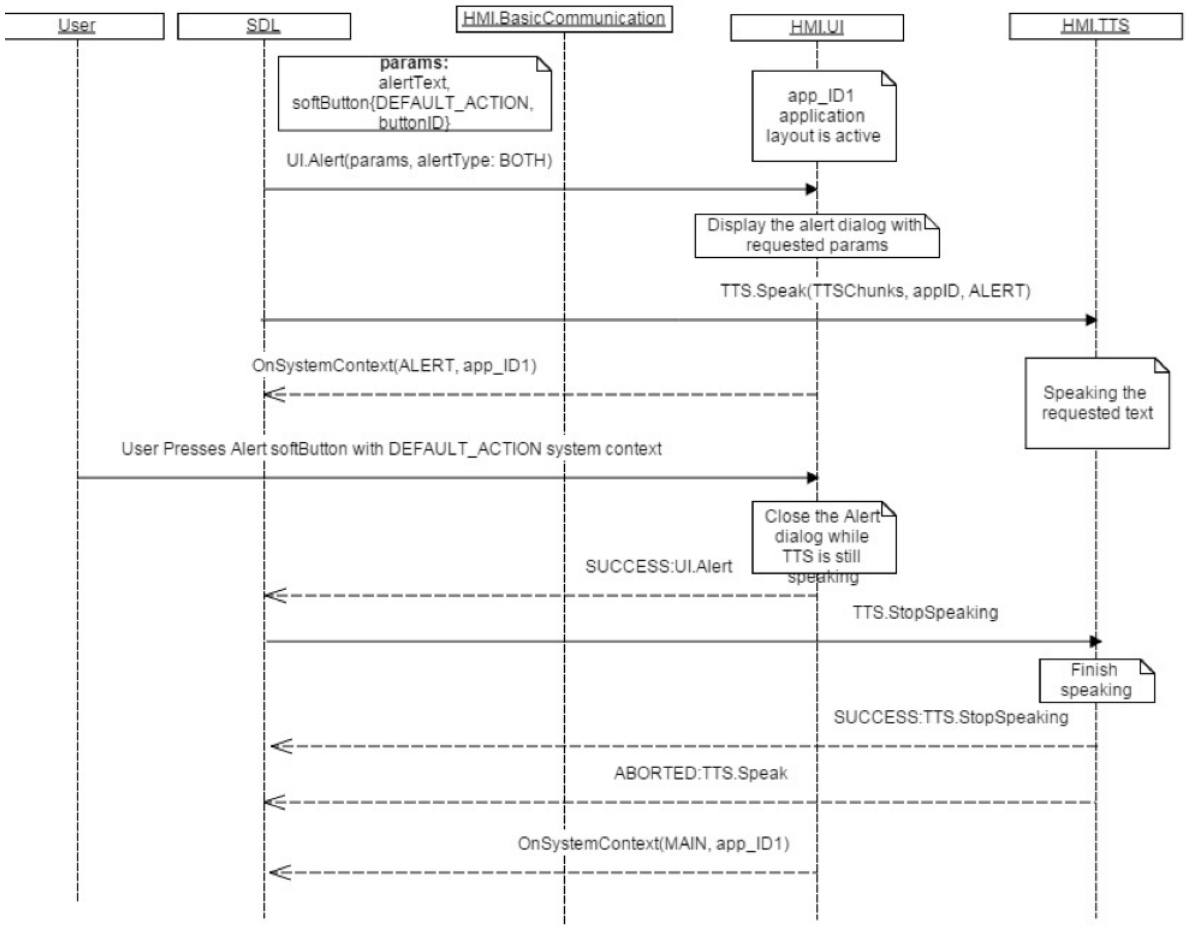
View Diagram



SEQUENCE DIAGRAM

Alert BOTH UI Closed before TTS finishes Speaking

View Diagram


User
SDL
HMI.BasicCommunication
HMI.UI
HMI.TTS
params:
alertText,
softButton{DEFAULT_ACTION,
buttonID}
app_ID1
application
layout is active
UI.Alert(params, alertType: BOTH)
Display the alert dialog with
requested params
TTS.Speak(TTSChunks, appID, ALERT)
OnSystemContext(ALERT, app_ID1)
Speaking the
requested text
User Presses Alert softButton with DEFAULT_ACTION system context
Close the Alert
dialog while
TTS is still
speaking
SUCCESS:UI.Alert
TTS.StopSpeaking
Finish
speaking
SUCCESS:TTS.StopSpeaking
ABORTED:TTS.Speak
OnSystemContext(MAIN, app_ID1)

Example Request

```
{
 "id" : 92,
 "jsonrpc" : "2.0",
 "method" : "UI. Alert",
 "params" :
 {
  "alertStrings" :
  [
    {
      "fieldName" : alertText1,
      "fieldText" : "WARNING"
    },
    {
      "fieldName" : alertText2,
      "fieldText" : "Hard weather conditions"
    }
  ],
  "duration" : 5000,
  "softButtons" :
  {
   "type" : TEXT,
   "text" : "OK",
   "softButtonID" : 697,
   "systemAction" : DEFAULT_ACTION
  },
   "alertType": "BOTH",
  "appID" : 65539
 }
}
```

Example Response


```
{
 "id" : 92,
 "jsonrpc" : "2.0",
 "result" :
 {
  "code" : 0,
  "method" : "UI.Alert"
 }
}
```


Example Error


```
{
 "id" : 92,
 "jsonrpc" : "2.0",
 "error" :
 {
  "code" : 4,
  "message" : "The requested command was rejected.",
  "data" :
  {
   "tryAgainTime" : 10000,
   "method" : "UI.Alert"
  }
 }
}
```

ChangeRegistration

Type Function
Sender SDL
Purpose

Change the display language for the specified application on the HMI

REQUEST

PARAMETERS

NAME	TYPE	MANDATORY	ADDITIONAL
appName	String	false	maxlength: 100
ngnMediaScreen AppName	String	false	maxlength: 100
language	Common.Language	true	
appHMIType	Common.AppHMIType	false	array: true minsize: 1 maxsize: 100
appID	Integer	true	

RESPONSE

PARAMETERS

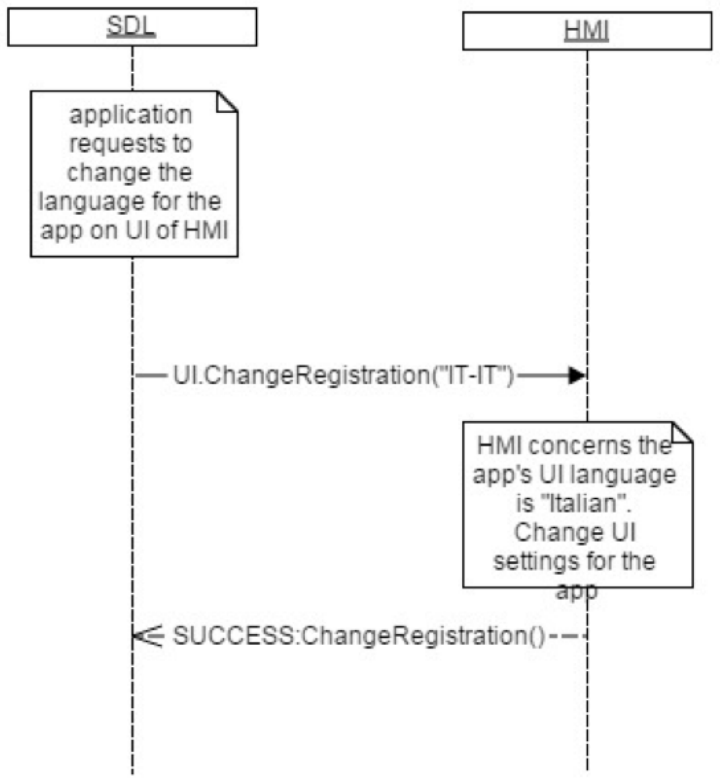
This RPC has no additional parameter requirements

Sequence Diagrams

SEQUENCE DIAGRAM

ChangeRegistration

View Diagram



Example Request

```
{
 "id" : 117,
 "jsonrpc" : "2.0",
 "method" : "UI.ChangeRegistration",
 "params" :
 {
  "Language" : "PT-PT",
  "appID" : 65146
 }
}
```

Example Response


```
{
 "id" : 117,
 "jsonrpc" : "2.0",
 "result" :
 {
  "code" : 0,
  "method" : "UI.ChangeRegistration"
 }
}
```


Example Error


```
{
 "id" : 117,
 "jsonrpc" : "2.0",
 "error" :
 {
  "code" : 11,
  "message" : "Invalid data",
  "data" :
  {
   "method" : "UI.ChangeRegistration"
  }
 }
}
```

ClosePopUp

Type Function
Sender SDL
Purpose Close the currently displayed popup UI element

REQUEST

PARAMETERS

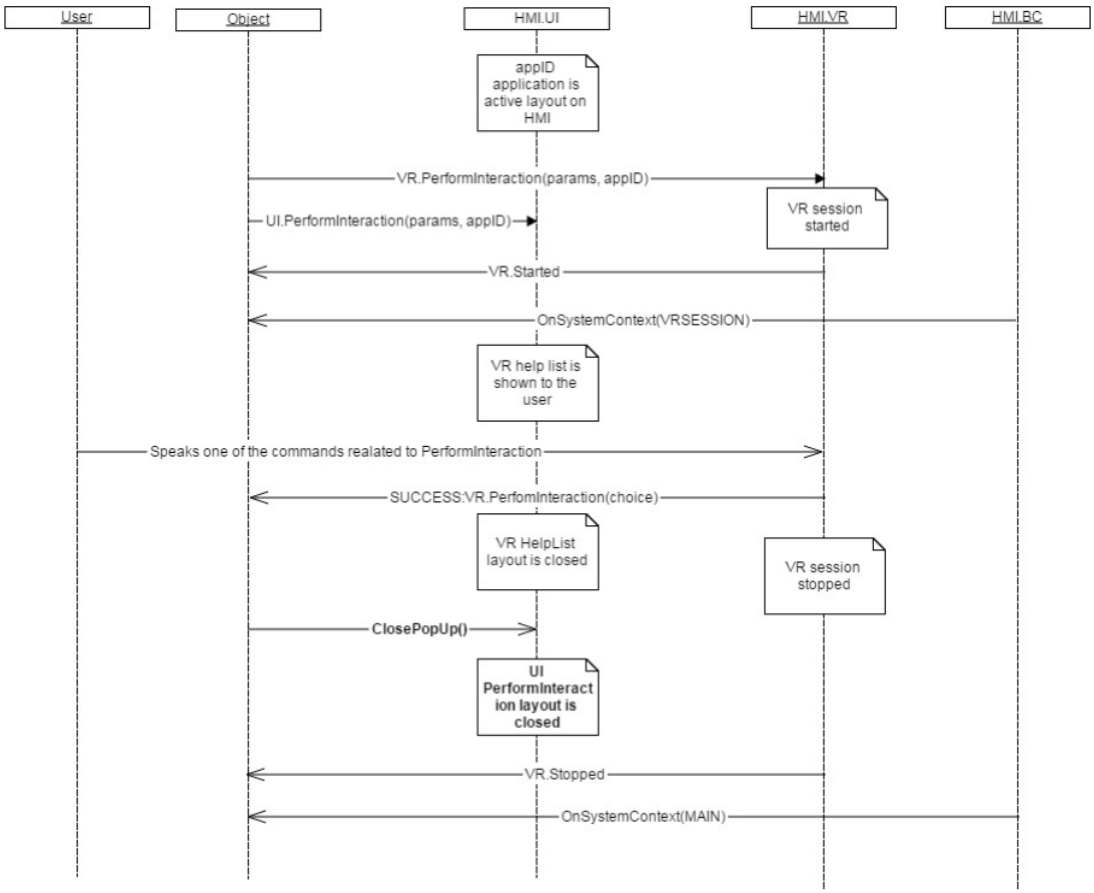
NAME	TYPE	MANDATORY	ADDITIONAL
methodName	String	false	

Sequence Diagrams

SEQUENCE DIAGRAM

ClosePopUp for UI.PerformInteraction

View Diagram



Example Request

```
{
 "id" : 79,
 "jsonrpc" : "2.0",
 "method" : "UI.ClosePopUp",
}
```

Example Response


```
{
 "id" : 79,
 "jsonrpc" : "2.0",
 "result" :
 {
  "code" : 0,
  "method" : "UI.ClosePopUp"
 }
}
```


Example Error


```
{
 "id" : 79,
 "jsonrpc" : "2.0",
 "error" :
 {
  "code" : 22,
  "message" : "During API call an unknown error has occurred",
  "data" :
  {
   "method" : "UI.ClosePopUp"
  }
 }
}
```

DeleteCommand

Type Function
Sender SDL
Purpose Delete a command from the specified application's menu or submenu

UI.DeleteCommand represents a request to remove a previously added command (added via UI.AddCommand) from the application's menu.

MUST

The application's menu must no longer display the command whose `cmdID` matches the RPC's `cmdID` when the user accesses the applications menu

REQUEST

PARAMETERS

NAME	TYPE	MANDATORY	ADDITIONAL
cmdID	Integer	true	minvalue: 0 maxvalue: 2000000000
appID	Integer	true	

RESPONSE

PARAMETERS

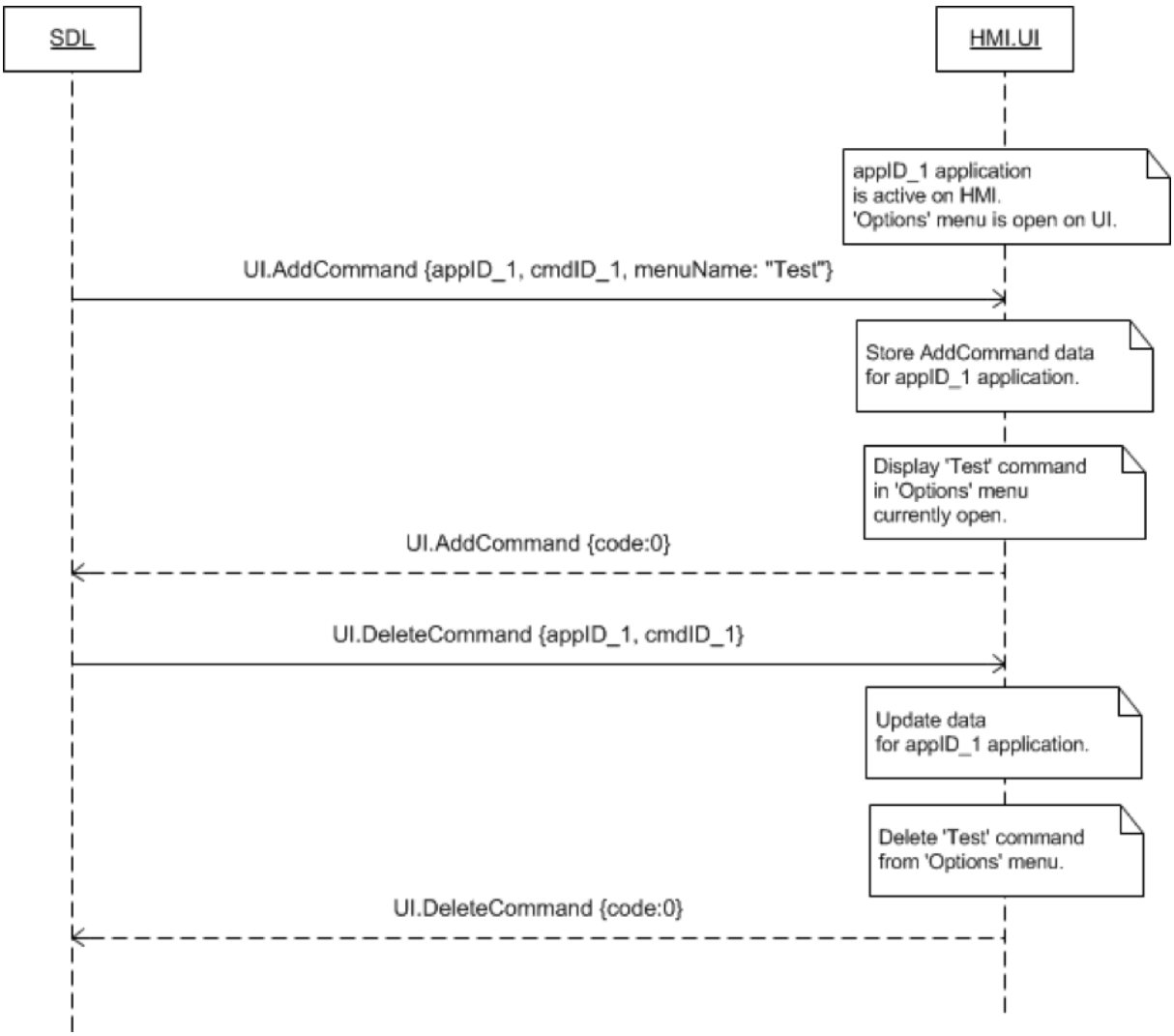
This RPC has no additional parameter requirements

Sequence Diagrams

SEQUENCE DIAGRAM

Delete Command with Menu Open

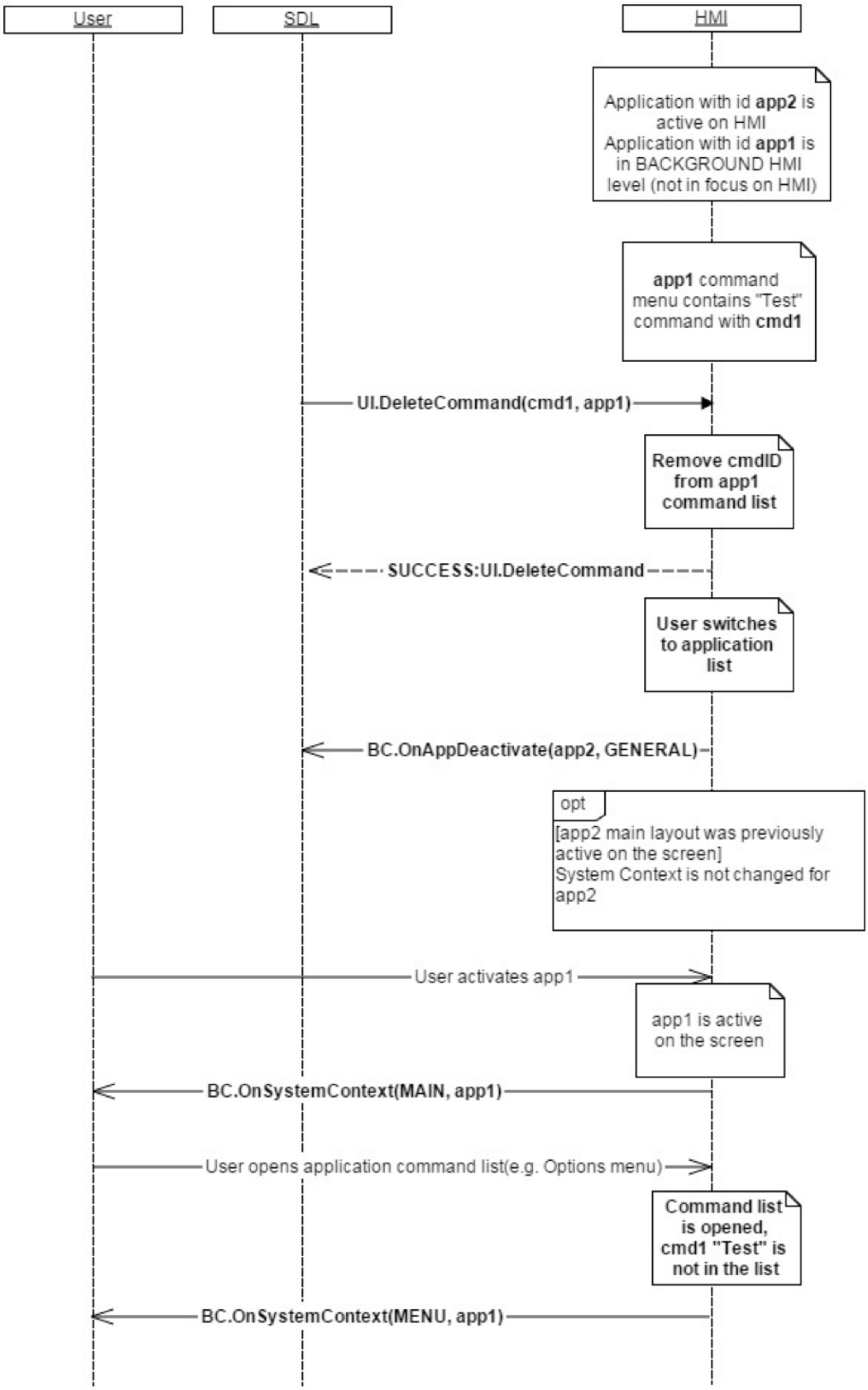
View Diagram



SEQUENCE DIAGRAM

Delete Command Application Inactive

View Diagram


User
SDL
HMI
Application with id app2 is active on HMI
Application with id app1 is in BACKGROUND HMI level (not in focus on HMI)
app1 command menu contains "Test" command with cmd1
UI.DeleteCommand(cmd1, app1)
Remove cmdID from app1 command list
SUCCESS:UI.DeleteCommand
User switches to application list
BC.OnAppDeactivate(app2, GENERAL)
opt
[app2 main layout was previously active on the screen]
System Context is not changed for app2
User activates app1
app1 is active on the screen
BC.OnSystemContext(MAIN, app1)
User opens application command list(e.g. Options menu)
Command list is opened, cmd1 "Test" is not in the list
BC.OnSystemContext(MENU, app1)

Example Request


```
{
 "id" : 70,
 "jsonrpc" : "2.0",
 "method" : "UI. DeleteCommand",
 "params" :
 {
  "cmdID" : 2318,
  "appID" : 65409
 }
}
```


Example Response


```
{
 "id" : 70,
 "jsonrpc" : "2.0",
 "result" :
 {
  "code" : 0,
  "method" : "UI. DeleteCommand"
 }
}
```

Example Error


```
{
 "id" : 70,
 "jsonrpc" : "2.0",
 "error" :
 {
  "code" : 13,
  "message" : "One of the provided IDs is not valid",
  "data" :
  {
   "UI. DeleteCommand"
  }
 }
}
```


DeleteSubMenu

Type Function
Sender SDL
Purpose Delete a sub menu from the specified application's menu

UI.DeleteSubMenu represents a request to remove a previously added sub menu (added via UI.AddSubMenu) from the application's menu.

> MUST
>
> The application's menu must no longer display the sub menu whose `menuID` matches the RPC's `menuID` when the user accesses the applications menu

REQUEST

PARAMETERS

NAME	TYPE	MANDATORY	ADDITIONAL
menuID	Integer	true	minvalue: 1 maxvalue: 2000000000
appID	Integer	true	

RESPONSE

PARAMETERS

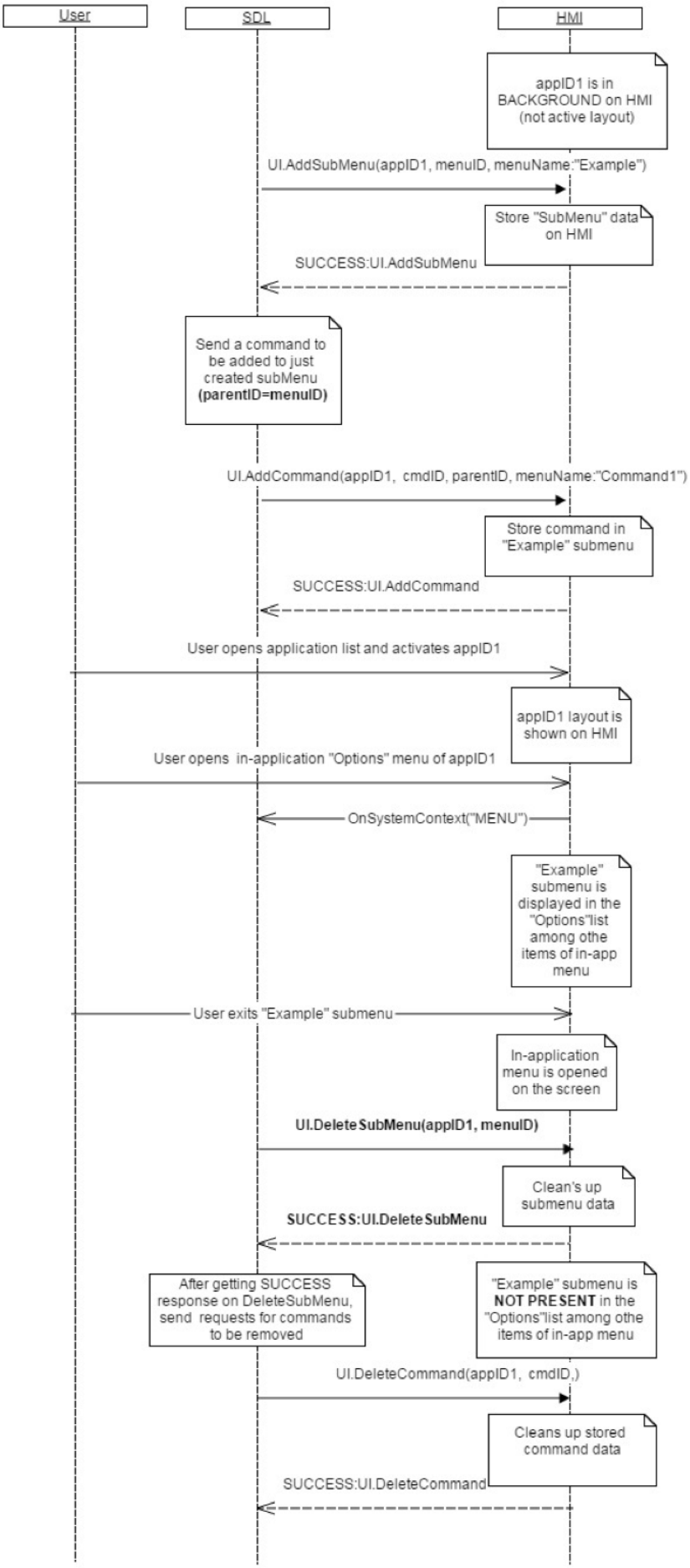
This RPC has no additional parameter requirements

Sequence Diagrams

SEQUENCE DIAGRAM

Delete Sub Menu Containing Commands

View Diagram


User
SDL
HMI
appID1 is in BACKGROUND on HMI (not active layout)
UI.AddSubMenu(appID1, menuID, menuName:"Example")
Store "SubMenu" data on HMI
SUCCESS:UI.AddSubMenu
Send a command to be added to just created subMenu (parentID=menuID)
UI.AddCommand(appID1, cmdID, parentID, menuName:"Command1")
Store command in "Example" submenu
SUCCESS:UI.AddCommand
User opens application list and activates appID1
appID1 layout is shown on HMI
User opens in-application "Options" menu of appID1
OnSystemContext("MENU")
"Example" submenu is displayed in the "Options"list among othe items of in-app menu
User exits "Example" submenu
In-application menu is opened on the screen
UI.DeleteSubMenu(appID1, menuID)
Clean's up submenu data
SUCCESS:UI.DeleteSubMenu
After getting SUCCESS response on DeleteSubMenu, send requests for commands to be removed
"Example" submenu is NOT PRESENT in the "Options"list among othe items of in-app menu
UI.DeleteCommand(appID1, cmdID,)
Cleans up stored command data
SUCCESS:UI.DeleteCommand

SEQUENCE DIAGRAM

Delete Sub Menu that is on screen

View Diagram

Example Request

```
{
 "id" : 70,
 "jsonrpc" : "2.0",
 "method" : "UI. DeleteSubMenu",
 "params" :
 {
  "menuID" : 345,
  "appID" : 65464
 }
}
```

Example Response

```
{
 "id" : 70,
 "jsonrpc" : "2.0",
 "result" :
 {
  "code" : 0,
  "method" : "UI. DeleteSubMenu"
 }
}
```

Example Error

```
{
  "id" : 70,
  "jsonrpc" : "2.0",
  "error" :
  {
    "code" : 8,
    "message" : "The data may not be changed because it is currently in
use",
    "data" :
    {
      "UI. DeleteSubMenu "
    }
  }
}
```

EndAudioPassThru

Type Function
Sender SDL
Purpose Stop audio capturing and sending PCM data to SDL

REQUEST

PARAMETERS

NAME	TYPE	MANDATORY	ADDITIONAL

RESPONSE

PARAMETERS

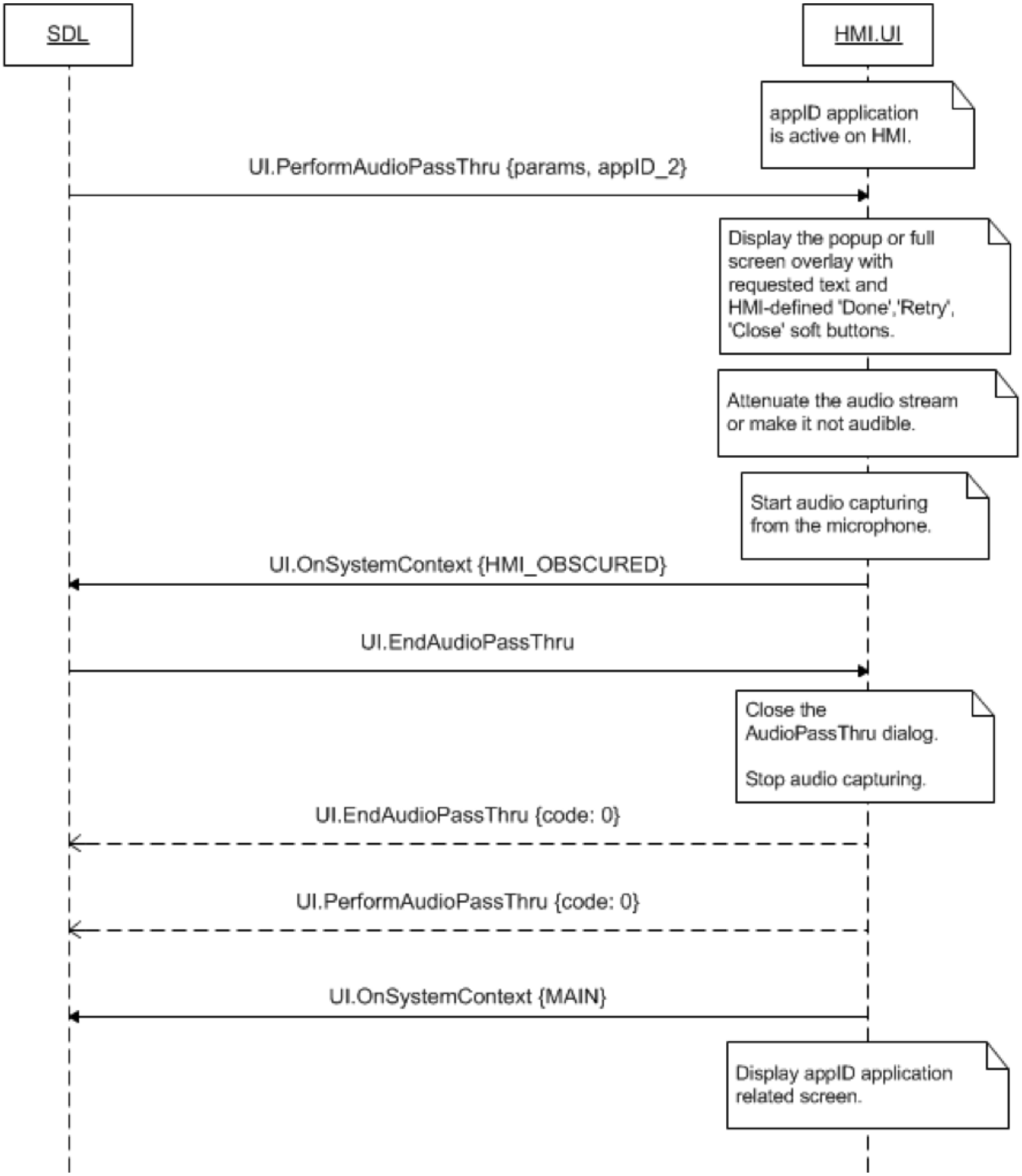
This RPC has no additional parameter requirements

Sequence Diagrams

SEQUENCE DIAGRAM

EndAudioPassThru stops audio capturing

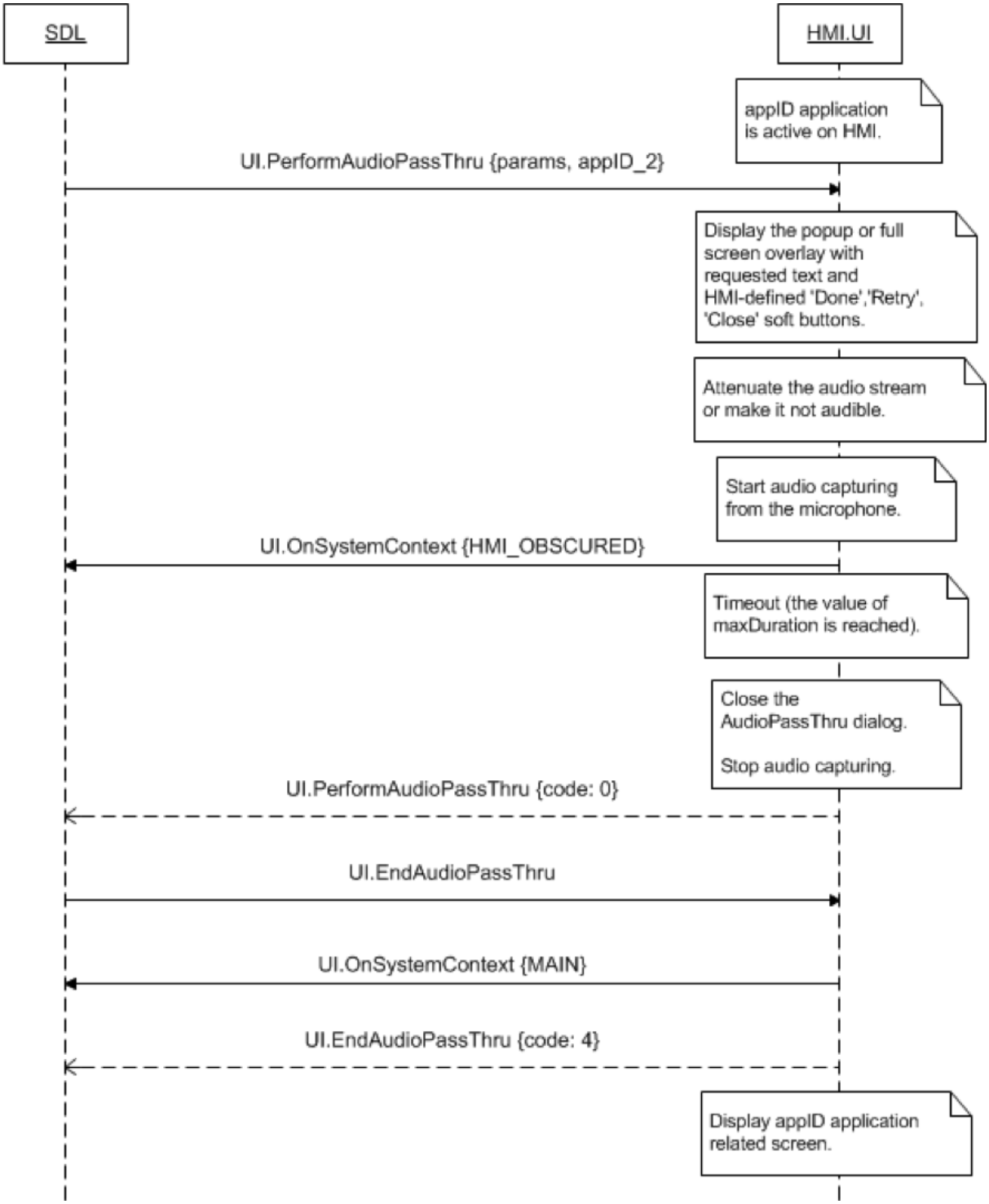
View Diagram



SEQUENCE DIAGRAM

EndAudioPassThru audio capturing already ended

View Diagram


SDL
HMI.UI
appID application is active on HMI.
UI.PerformAudioPassThru {params, appID_2}
Display the popup or full screen overlay with requested text and HMI-defined 'Done','Retry', 'Close' soft buttons.
Attenuate the audio stream or make it not audible.
Start audio capturing from the microphone.
UI.OnSystemContext {HMI_OBSCURED}
Timeout (the value of maxDuration is reached).
Close the AudioPassThru dialog.
Stop audio capturing.
UI.PerformAudioPassThru {code: 0}
UI.EndAudioPassThru
UI.OnSystemContext {MAIN}
UI.EndAudioPassThru {code: 4}
Display appID application related screen.

Example Request

```
{
 "id" : 79,
 "jsonrpc" : "2.0",
 "method" : "UI.EndAudioPassThru",
}
```

Example Response

```
{
 "id" : 79,
 "jsonrpc" : "2.0",
 "result" :
 {
  "code" : 0,
  "method" : "UI.EndAudioPassThru"
 }
}
```

Example Error


```
{
 "id" : 79,
 "jsonrpc" : "2.0",
 "error" :
 {
  "code" : 4,
  "message" : "Rejected: no PerformAudioPassThru is now active",
  "data" :
  {
   "method" : "UI.EndAudioPassThru"
  }
 }
}
```


GetCapabilities

Type Function
Sender SDL
Purpose Inform SDL of the UI capabilities of the vehicle.

REQUEST

PARAMETERS

NAME	TYPE	MANDATORY	ADDITIONAL

RESPONSE

PARAMETERS

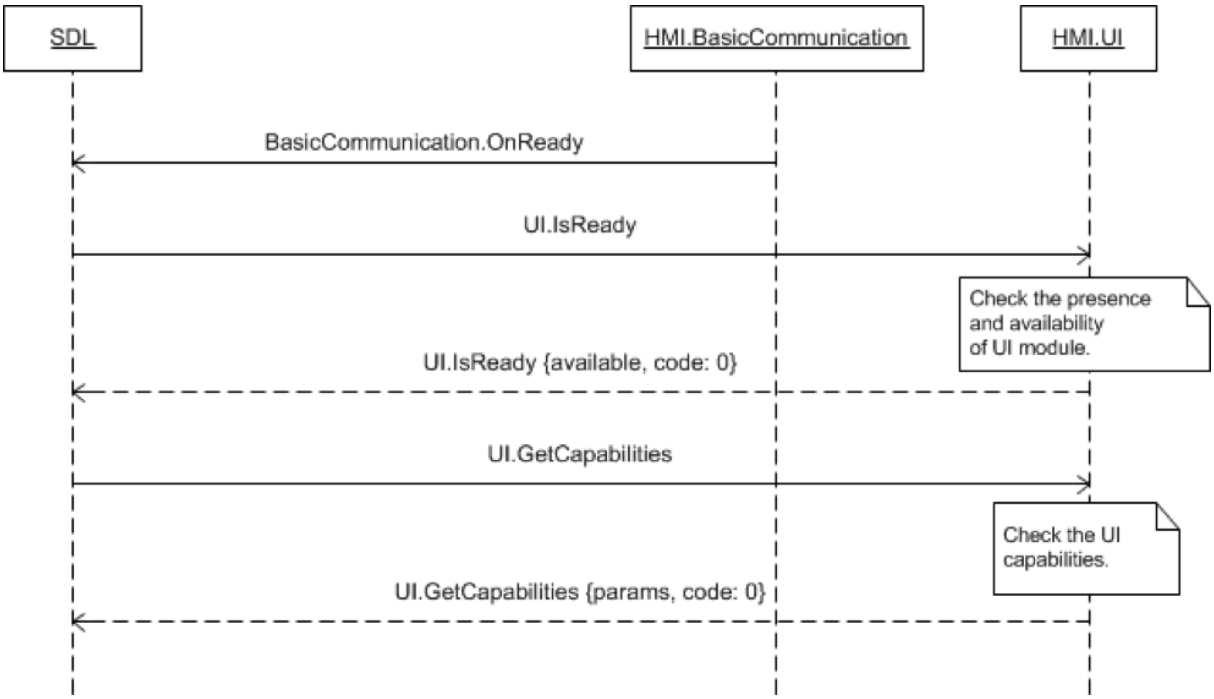
NAME	TYPE	MANDATORY	ADDITIONAL
displayCapabilities	Common.DisplayCapabilities	true	
audioPassThruCapabilities	Common.AudioPassThruCapabilities	true	
hmiZoneCapabilities	Common.HmiZoneCapabilities	true	
softButtonCapabilities	Common.SoftButtonCapabilities	false	array: true minsize: 1 maxsize: 100
hmiCapabilities	Common.HMICapabilities	false	
systemCapabilities	Common.SystemCapabilities	false	

Sequence Diagrams

SEQUENCE DIAGRAM

Get Capabilities

View Diagram



Example Request

```
{
  "id" : 18,
  "jsonrpc" : "2.0",
  "method" : "UI.GetCapabilities"
}
```

Example Response

```
{
  "id" : 18,
  "jsonrpc" : "2.0",
  "result" :
  {
    "displayCapabilities" :
    {
      "displayType" : "GEN2_8_DMA",
      "textFields" : ["mainField1", "mainField2", "mediaclock",
"mediaTrack", "alertText1", "alertText2", "alertText3",
"scrollableMessageBody", "initialInteractionText", "navigationText1",
"navigationText2", "audioPassThruDisplayText1",
"audioPassThruDisplayText2", "notificationText"],
      "mediaClockFormats" : ["CLOCK1", "CLOCKTEXT4"],
      "graphicSupported" : true,
      "imageCapabilities": ["DYNAMIC"]
    },
    "hmiCapabilities" :
    {
      "navigation" : true,
      "phoneCall" : true,
      "videostreaming" : true
    },
    "systemCapabilities":
    {
      "navigationCapability":
      {
        "sendLocationEnabled": true,
        "getWayPointsEnabled": true
      },
      "phoneCapability":
      {
        "dialNumberEnabled": true
      },
      "videoStreamingCapability":
      {
        "preferredResolution": {
          "resolutionWidth": 800,
          "resolutionHeight": 350
        },
        "maxBitrate": 10000,
        "supportedFormats": [{
          "protocol": "RAW",
          "codec": "H264"
        }],
        "hapticSpatialDataSupported": true
      }
    },
    "softButtonCapabilities" :
```

```
    [
      {
        "shortPressAvailable" : true,
        "longPressAvailable" : true,
        "upDownAvailable" : true,
        "imageSupported" : true
      }
    ],
    "hmiZoneCapabilities" : "FRONT",
    "audioPassThruCapabilities" :
    {
      "samplingRate" : "44KHZ",
      "bitsPerSample" : "8_BIT",
      "audioType" : "PCM"
    },
    "code" : 0,
    "method" : "UI.GetCapabilities"
  }
}
```

Example Error

```
{
  "id" : 18,
  "jsonrpc" : "2.0",
  "error" :
  {
    "code" : 22,
    "message" : "During API call the unknown error has occurred",
    "data" :
    {
      "method" : "UI.GetCapabilities"
    }
  }
}
```

GetLanguage

Type

Function

Sender

SDL

Purpose

Inform SDL of the current HMI display language

REQUEST

PARAMETERS

NAME	TYPE	MANDATORY	ADDITIONAL

RESPONSE

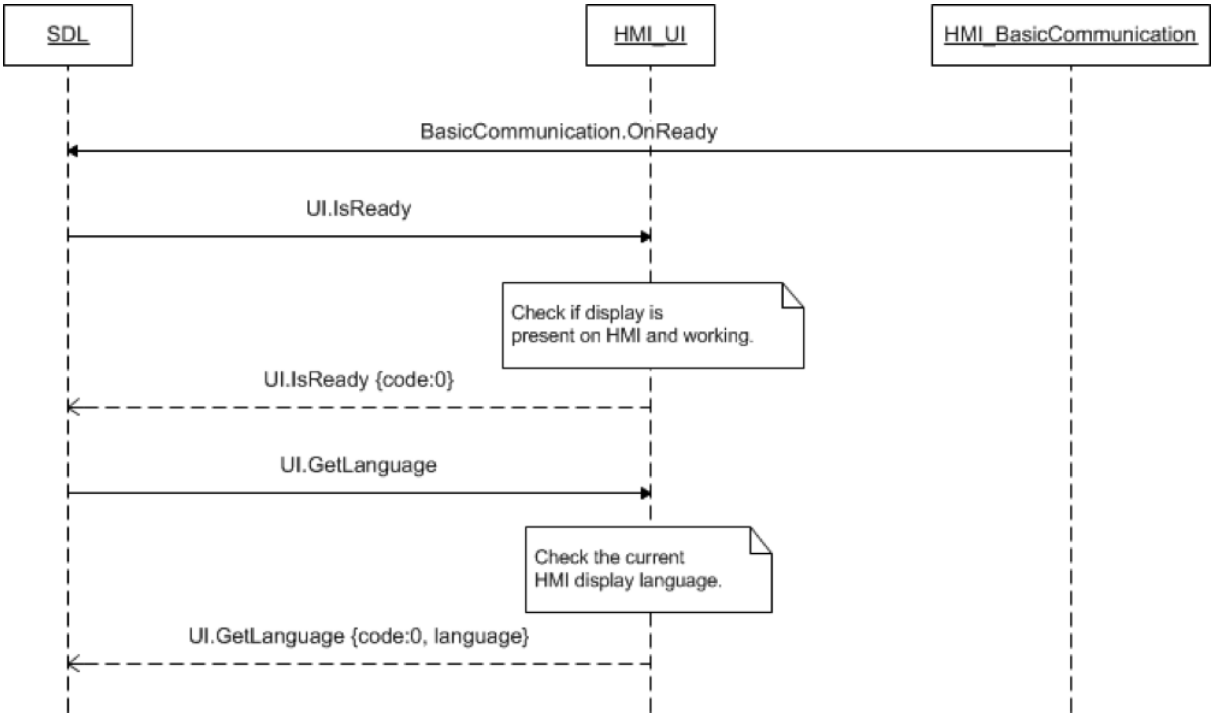
PARAMETERS

NAME	TYPE	MANDATORY	ADDITIONAL
language	Common.Language	true	
### Sequence Diagrams			

SEQUENCE DIAGRAM

GetLanguage

View Diagram



Example Request

```
{
 "id" : 167,
 "jsonrpc" : "2.0",
 "method" : "UI.GetLanguage"
}
```

Example Response

```
{
 "id" : 167,
 "jsonrpc" : "2.0",
 "result" :
 {
  "language" : "ES-ES",
  "code" : 0,
  "method" : "UI.GetLanguage"
 }
}
```

Example Error


```
{
 "id" : 167,
 "jsonrpc" : "2.0",
 "error" :
 {
  "code" : 11,
  "message" : "Invalid data",
  "data" :
  {
   "method" : "UI.GetLanguage"
  }
 }
}
```


GetSupportedLanguages

Type Function
Sender SDL
Purpose Get the supported UI languages

REQUEST

PARAMETERS

NAME	TYPE	MANDATORY	ADDITIONAL

RESPONSE

PARAMETERS

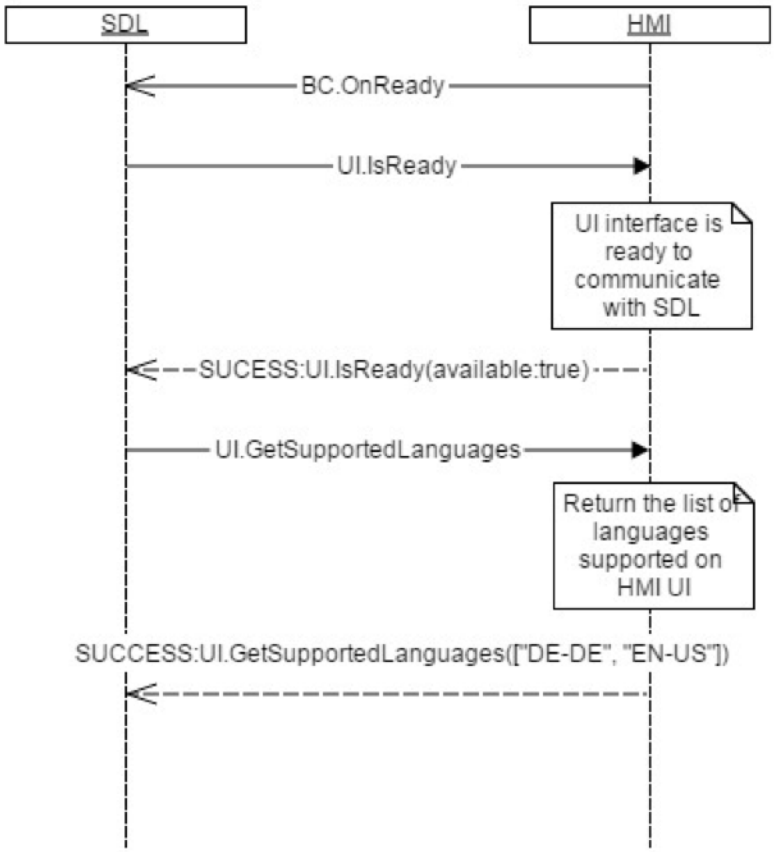
NAME	TYPE	MANDATORY	ADDITIONAL
languages	Common.Language	true	array: true minsize: 1 maxsize: 100

Sequence Diagrams

SEQUENCE DIAGRAM

Get Supported Languages

View Diagram



Example Request

```
{
  "id" : 99,
  "jsonrpc" : "2.0",
  "method" : "UI.GetSupportedLanguages"
}
```

Example Response


```
{
 "id" : 99,
 "jsonrpc" : "2.0",
 "result" :
 {
  "languages" : ["AR-SA", "DE-DE", "EN-GB", "EN-US", "ES-ES", "FR-FR",
"IT-IT"],
  "code" : 0,
  "method" : "UI.GetSupportedLanguages"
 }
}
```


Example Error


```
{
 "id" : 99,
 "jsonrpc" : "2.0",
 "error" :
 {
  "code" : 9,
  "message" : "The requested data is not available",
  "data" :
  {
   "method" : "UI.GetSupportedLanguages"
  }
 }
}
```

IsReady

Type

Function

Sender

SDL

Purpose

Request ready state of UI Module

REQUEST

NOTE

1. SDL sends `UI.IsReady` request after HMI confirms its readiness via `BC.OnReady` notification.
2. If HMI responds with `"available":false`, SDL will not further communicate over UI interface with HMI.
3. If HMI does not respond during SDL's default timeout, SDL will continue to send RPCs over UI interface to HMI.

PARAMETERS

The request does not have specific parameters.

RESPONSE

MUST

1. Check whether UI component is available and ready.
2. Respond correspondingly to results of this check.

PARAMETERS

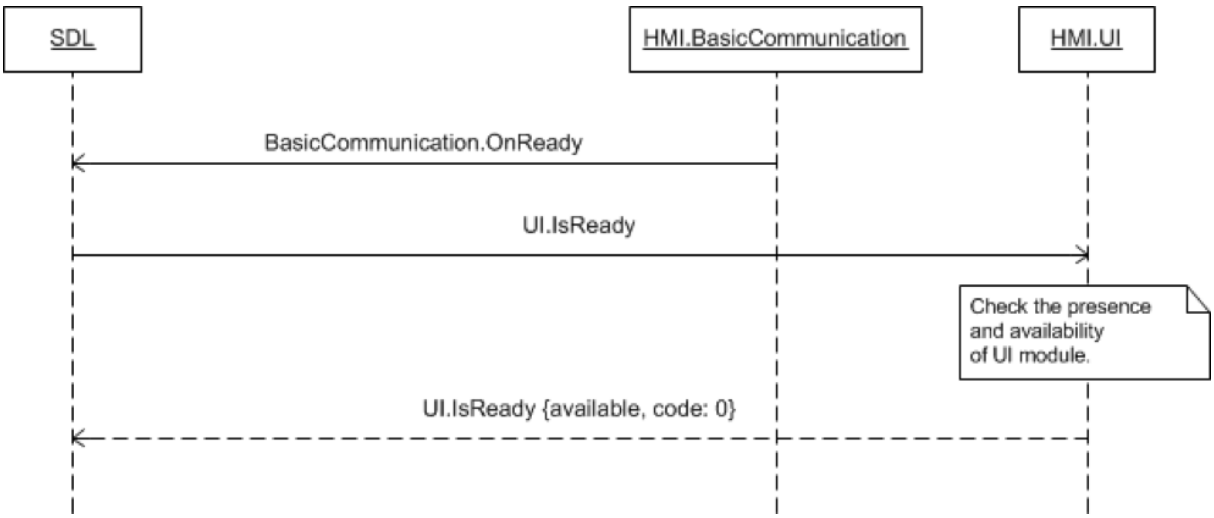
NAME	TYPE	MANDATORY	ADDITIONAL
available	Boolean	true	

Sequence Diagrams

SEQUENCE DIAGRAM

UI Component Ready

View Diagram



Example Request

```
{
 "id" : 8,
 "jsonrpc" : "2.0",
 "method" : "UI.IsReady"
}
```

Example Response

```
{
 "id" : 8,
 "jsonrpc" : "2.0",
 "result" :
 {
  "available" : false,
  "code" : 0,
  "method" : "UI.IsReady"
 }
}
```

Example Error

```
{
  "id" : 8,
  "jsonrpc" : "2.0",
  "error" :
  {
    "code" : 22,
    "message" : " HMI doesn’t have the information about UI availability
or some failure occurred ",
    "data" :
    {
      "method" : "UI.IsReady"
    }
  }
}
```

OnCommand

Type Notification
Sender HMI
Purpose Inform SDL that a command has been chosen by the User from the UI.

A notification to inform SDL about a command being selected which had previously been added via UI.AddCommand

MUST

The HMI must send the UI.OnCommand notification to SDL when the user selects a command from the application's menu or sub menu.

Notification

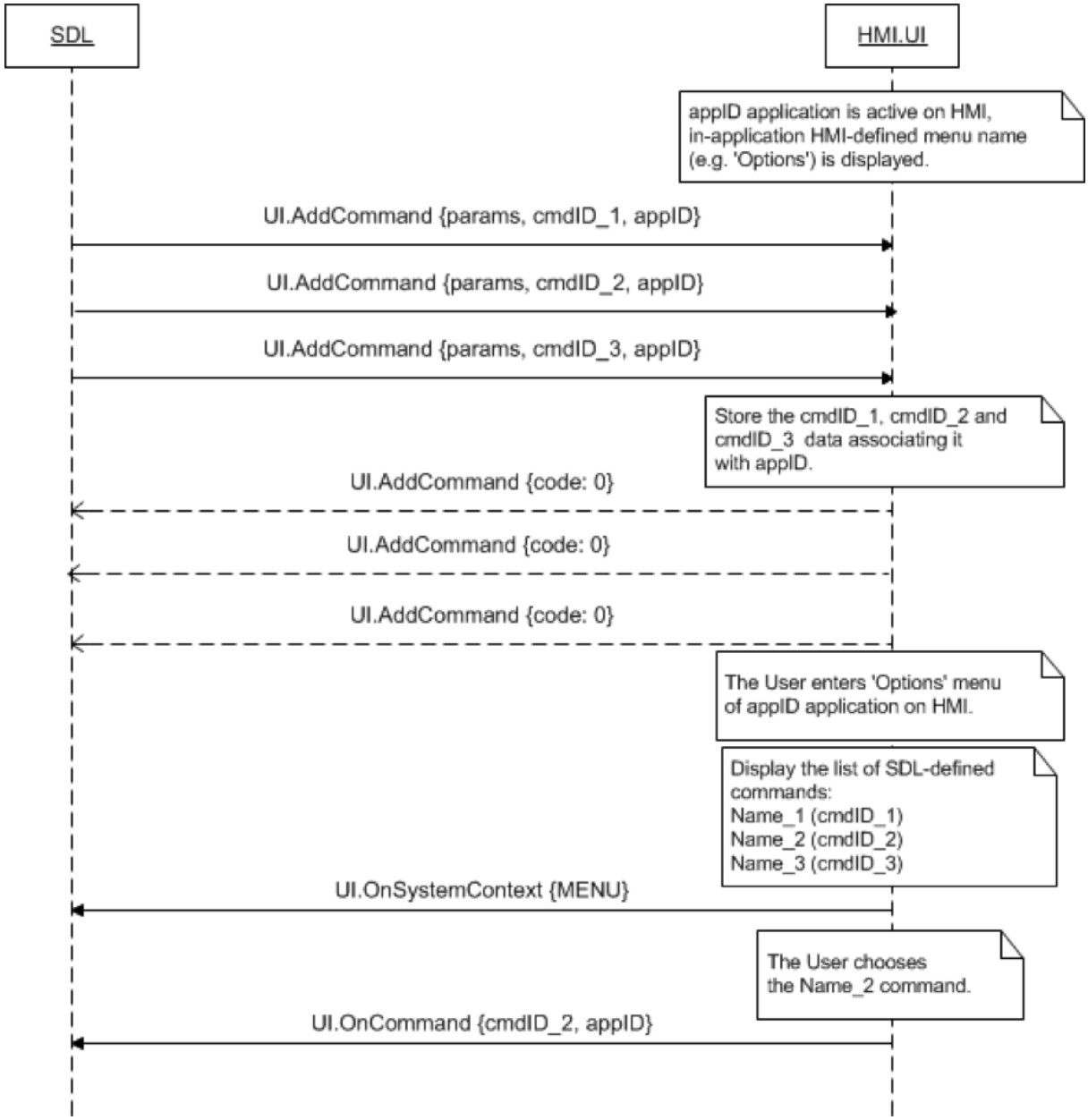
PARAMETERS

NAME	TYPE	MANDATORY	ADDITIONAL
cmdID	Integer	true	minvalue: 0 maxvalue: 2000000000
appID	Integer	true	
### Sequence Diagrams			

SEQUENCE DIAGRAM

OnCommand

View Diagram


SDL
HMI.UI
appID application is active on HMI,
in-application HMI-defined menu name
(e.g. 'Options') is displayed.
UI.AddCommand {params, cmdID_1, appID}
UI.AddCommand {params, cmdID_2, appID}
UI.AddCommand {params, cmdID_3, appID}
Store the cmdID_1, cmdID_2 and
cmdID_3 data associating it
with appID.
UI.AddCommand {code: 0}
UI.AddCommand {code: 0}
UI.AddCommand {code: 0}
The User enters 'Options' menu
of appID application on HMI.
Display the list of SDL-defined
commands:
Name_1 (cmdID_1)
Name_2 (cmdID_2)
Name_3 (cmdID_3)
UI.OnSystemContext {MENU}
The User chooses
the Name_2 command.
UI.OnCommand {cmdID_2, appID}

JSON EXAMPLE NOTIFICATION

```
{
 "jsonrpc" : "2.0",
 "method" : "UI.OnCommand",
 "params" :
 {
  "cmdID" : 2318,
   "appID" : 65409
 }
}
```

OnDriverDistraction

Type Notification
Sender HMI
Purpose Inform SDL about changes to the Driver Distraction state.

Notification

PARAMETERS

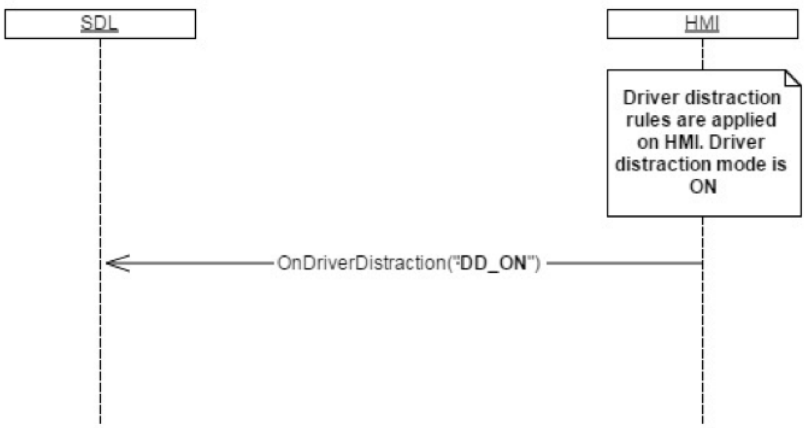
NAME	TYPE	MANDATORY	ADDITIONAL
state	Common.DriverDistractionState	true	

Sequence Diagrams

SEQUENCE DIAGRAM

OnDriverDistraction

View Diagram



JSON EXAMPLE NOTIFICATION

```
{
 "jsonrpc" : "2.0",
 "method" : "UI.OnDriverDistraction",
 "params" :
 {
  "state" : "DD_ON"
 }
}
```

OnKeyboardInput

Type Notification
Sender HMI
Purpose

Inform SDL that a keyboard event has occurred.

Notification

PARAMETERS

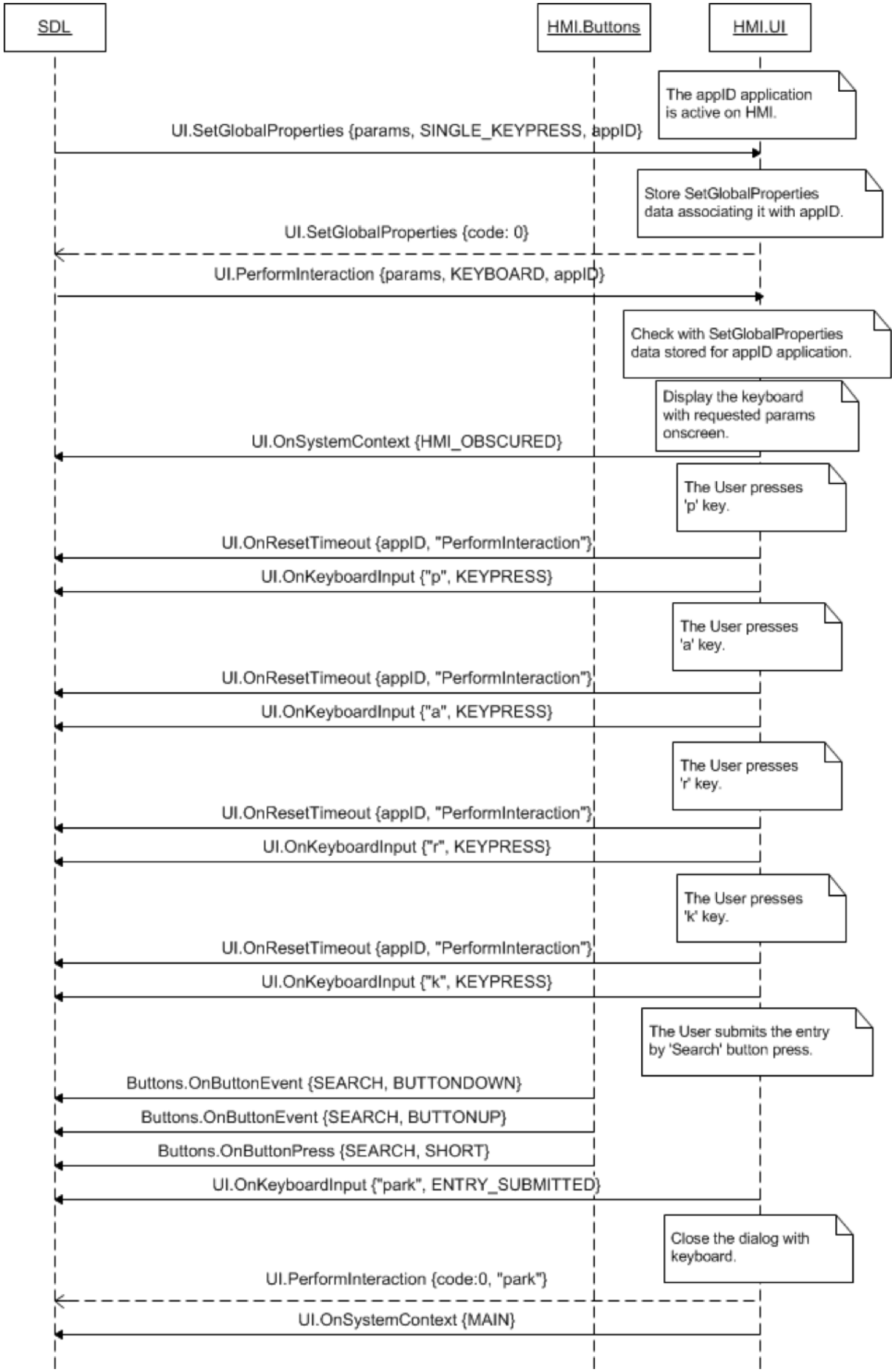
NAME	TYPE	MANDATORY	ADDITIONAL
event	Common.KeyboardEvent	true	
data	String	false	minlength: 0 maxlength: 500

Sequence Diagrams

SEQUENCE DIAGRAM

OnKeyboardInput SINGLE_KEYPRESS mode

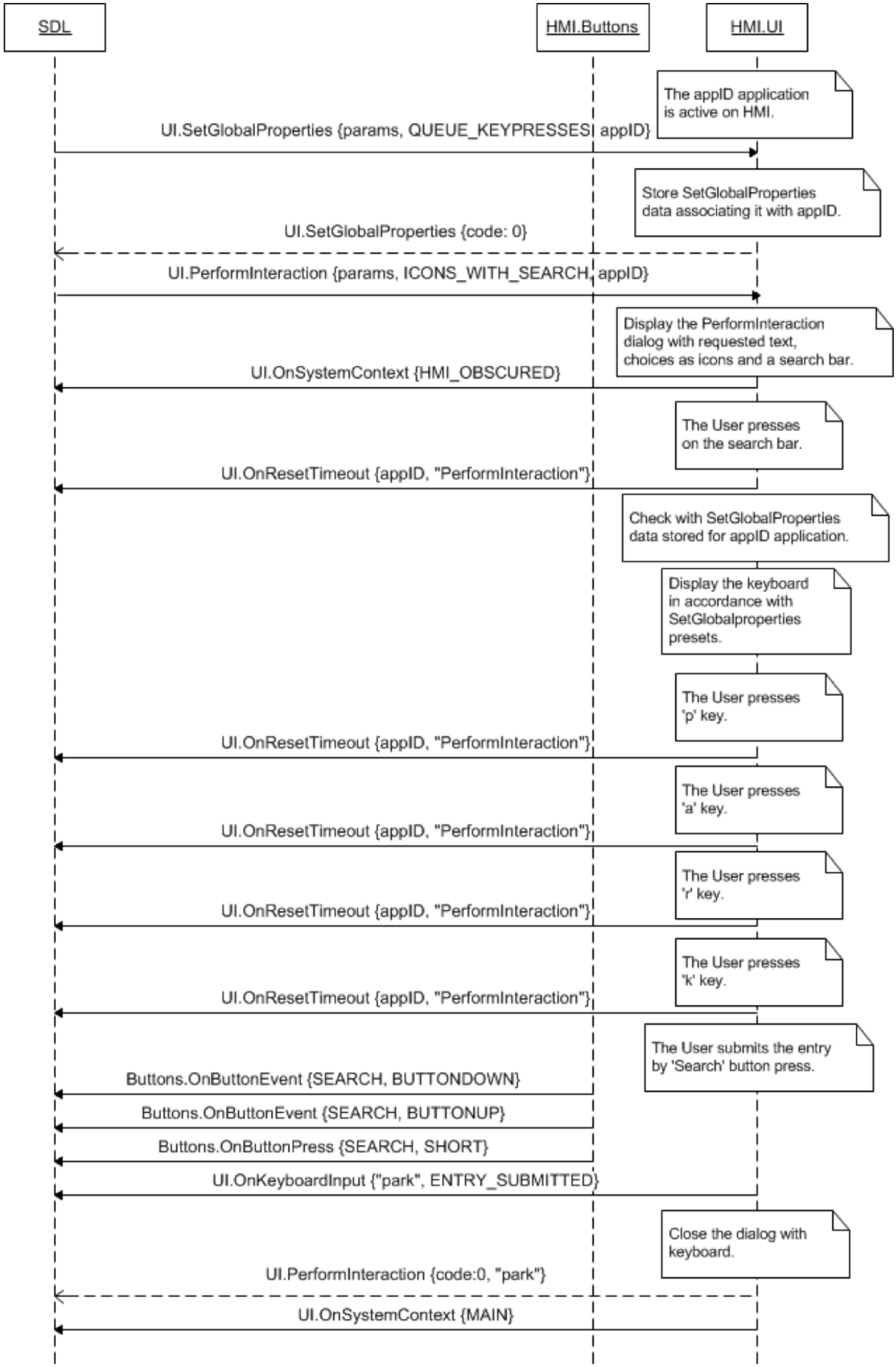
View Diagram


SDL
HMI.Buttons
HMI.UI
The appID application is active on HMI.
UI.SetGlobalProperties {params, SINGLE_KEYPRESS, appID}
Store SetGlobalProperties data associating it with appID.
UI.SetGlobalProperties {code: 0}
UI.PerformInteraction {params, KEYBOARD, appID}
Check with SetGlobalProperties data stored for appID application.
Display the keyboard with requested params onscreen.
UI.OnSystemContext {HMI_OBSCURED}
The User presses 'p' key.
UI.OnResetTimeout {appID, "PerformInteraction"}
UI.OnKeyboardInput {"p", KEYPRESS}
The User presses 'a' key.
UI.OnResetTimeout {appID, "PerformInteraction"}
UI.OnKeyboardInput {"a", KEYPRESS}
The User presses 'r' key.
UI.OnResetTimeout {appID, "PerformInteraction"}
UI.OnKeyboardInput {"r", KEYPRESS}
The User presses 'k' key.
UI.OnResetTimeout {appID, "PerformInteraction"}
UI.OnKeyboardInput {"k", KEYPRESS}
The User submits the entry by 'Search' button press.
Buttons.OnButtonEvent {SEARCH, BUTTONDOWN}
Buttons.OnButtonEvent {SEARCH, BUTTONUP}
Buttons.OnButtonPress {SEARCH, SHORT}
UI.OnKeyboardInput {"park", ENTRY_SUBMITTED}
Close the dialog with keyboard.
UI.PerformInteraction {code:0, "park"}
UI.OnSystemContext {MAIN}

SEQUENCE DIAGRAM

OnKeyboardInput QUEUE_KEYPRESSES mode

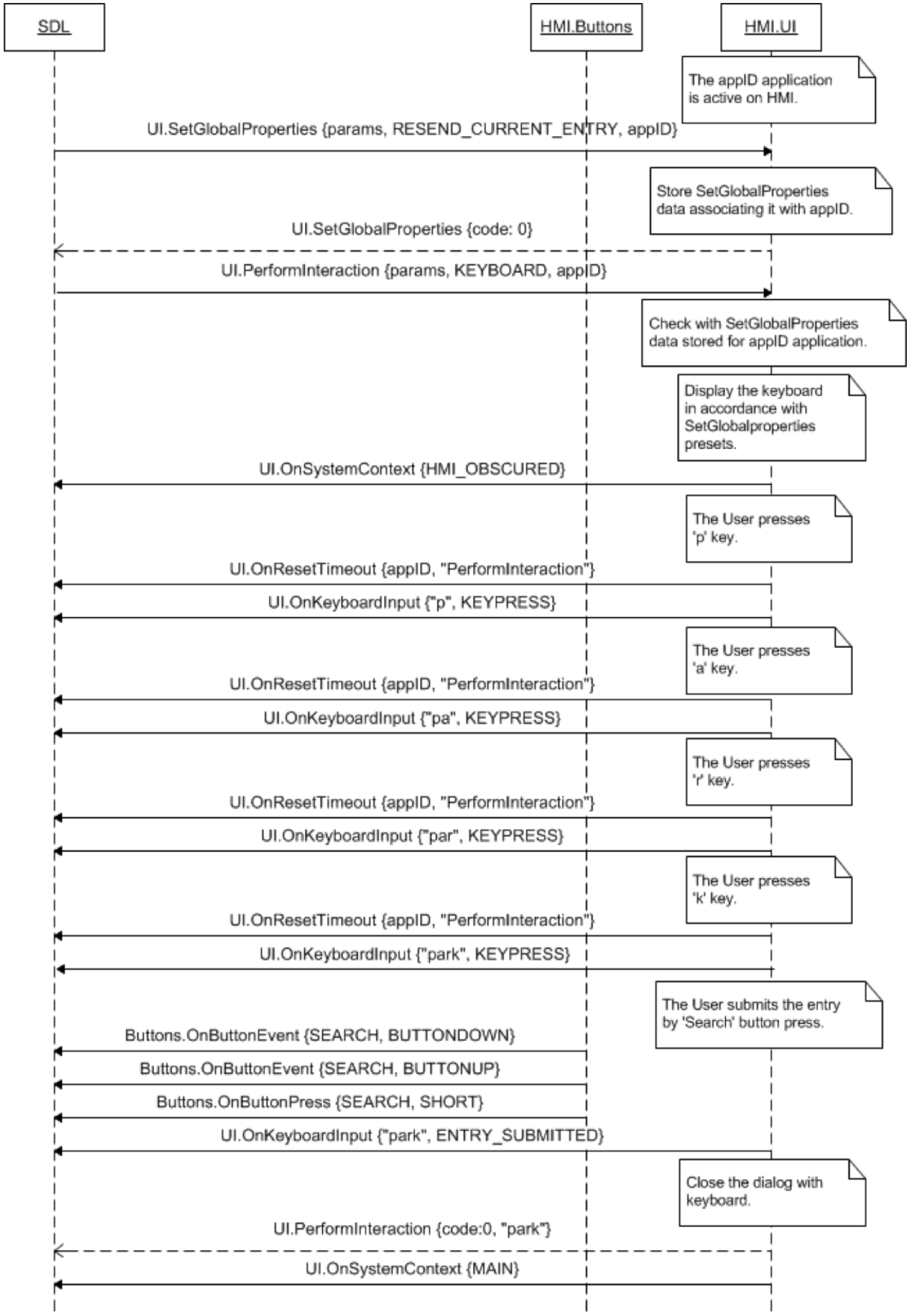
View Diagram


SDL
HMI.Buttons
HMI.UI
The appID application is active on HMI.
UI.SetGlobalProperties {params, QUEUE_KEYPRESSES, appID}
Store SetGlobalProperties data associating it with appID.
UI.SetGlobalProperties {code: 0}
UI.PerformInteraction {params, ICONS_WITH_SEARCH, appID}
Display the PerformInteraction dialog with requested text, choices as icons and a search bar.
UI.OnSystemContext {HMI_OBSCURED}
The User presses on the search bar.
UI.OnResetTimeout {appID, "PerformInteraction"}
Check with SetGlobalProperties data stored for appID application.
Display the keyboard in accordance with SetGlobalproperties presets.
The User presses 'p' key.
UI.OnResetTimeout {appID, "PerformInteraction"}
The User presses 'a' key.
UI.OnResetTimeout {appID, "PerformInteraction"}
The User presses 'r' key.
UI.OnResetTimeout {appID, "PerformInteraction"}
The User presses 'k' key.
UI.OnResetTimeout {appID, "PerformInteraction"}
The User submits the entry by 'Search' button press.
Buttons.OnButtonEvent {SEARCH, BUTTONDOWN}
Buttons.OnButtonEvent {SEARCH, BUTTONUP}
Buttons.OnButtonPress {SEARCH, SHORT}
UI.OnKeyboardInput {"park", ENTRY_SUBMITTED}
Close the dialog with keyboard.
UI.PerformInteraction {code:0, "park"}
UI.OnSystemContext {MAIN}

SEQUENCE DIAGRAM

OnKeyboardInput RESEND_CURRENT_ENTRY mode

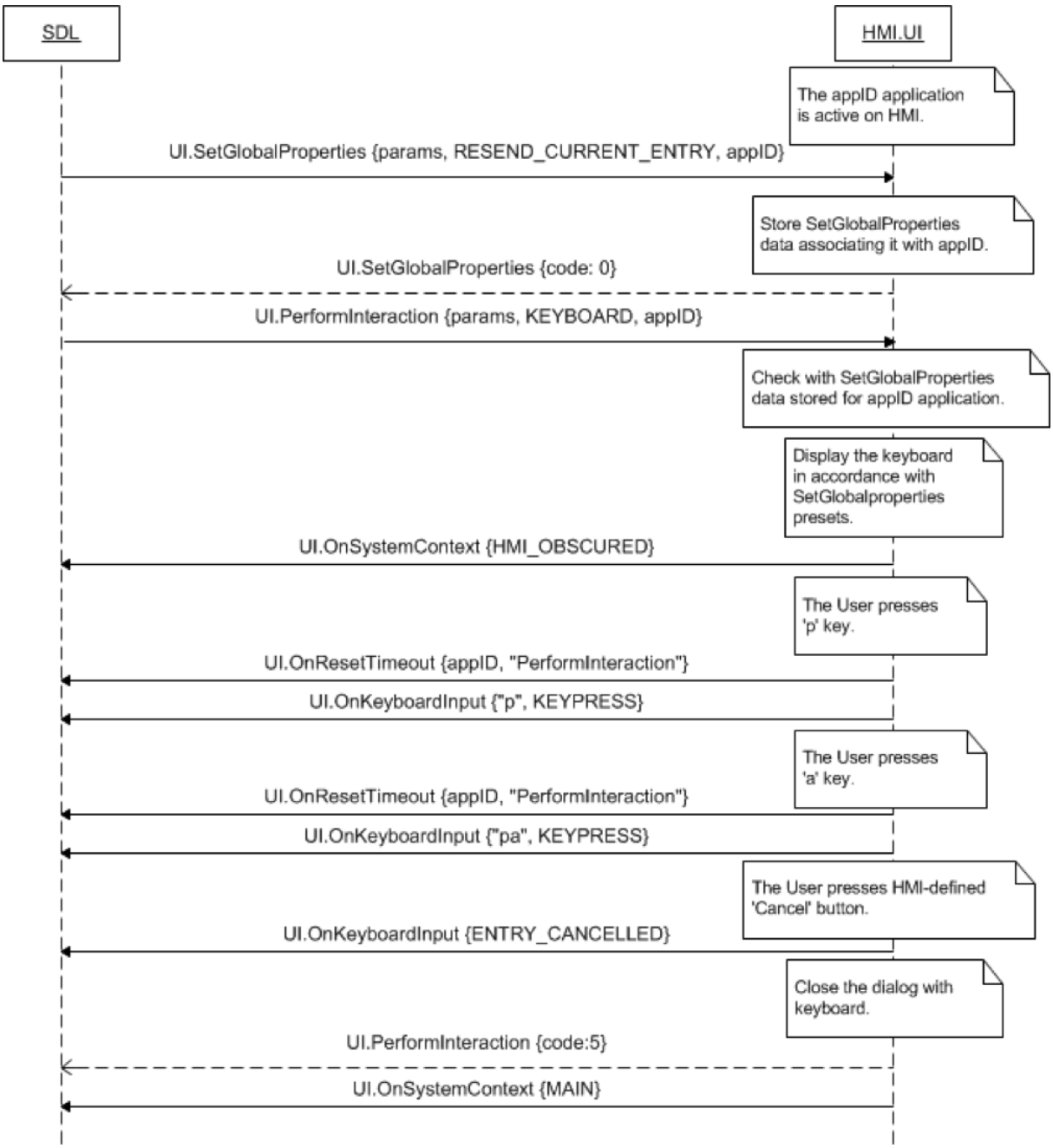
View Diagram


SDL
HMI.Buttons
HMI.UI
The appID application is active on HMI.
UI.SetGlobalProperties {params, RESEND_CURRENT_ENTRY, appID}
Store SetGlobalProperties data associating it with appID.
UI.SetGlobalProperties {code: 0}
UI.PerformInteraction {params, KEYBOARD, appID}
Check with SetGlobalProperties data stored for appID application.
Display the keyboard in accordance with SetGlobalproperties presets.
UI.OnSystemContext {HMI_OBSCURED}
The User presses 'p' key.
UI.OnResetTimeout {appID, "PerformInteraction"}
UI.OnKeyboardInput {"p", KEYPRESS}
The User presses 'a' key.
UI.OnResetTimeout {appID, "PerformInteraction"}
UI.OnKeyboardInput {"pa", KEYPRESS}
The User presses 'r' key.
UI.OnResetTimeout {appID, "PerformInteraction"}
UI.OnKeyboardInput {"par", KEYPRESS}
The User presses 'k' key.
UI.OnResetTimeout {appID, "PerformInteraction"}
UI.OnKeyboardInput {"park", KEYPRESS}
The User submits the entry by 'Search' button press.
Buttons.OnButtonEvent {SEARCH, BUTTONDOWN}
Buttons.OnButtonEvent {SEARCH, BUTTONUP}
Buttons.OnButtonPress {SEARCH, SHORT}
UI.OnKeyboardInput {"park", ENTRY_SUBMITTED}
Close the dialog with keyboard.
UI.PerformInteraction {code:0, "park"}
UI.OnSystemContext {MAIN}

SEQUENCE DIAGRAM

OnKeyboardInput cancelled

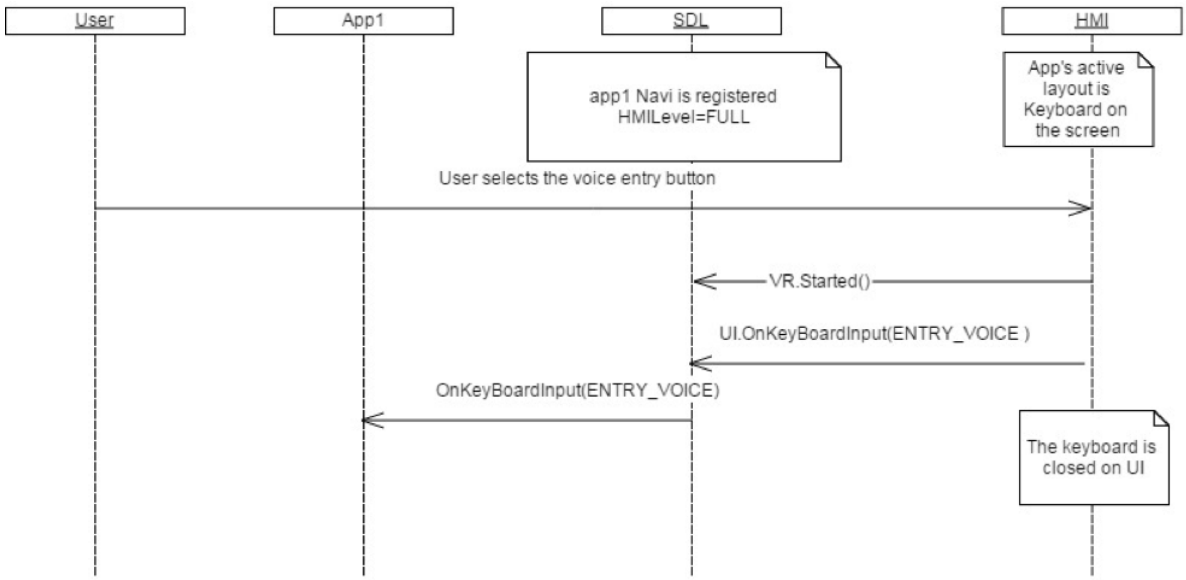
View Diagram



SEQUENCE DIAGRAM

OnKeyboardInput ENTRY_VOICE mode

View Diagram



JSON EXAMPLE NOTIFICATION

```
{
 "jsonrpc" : "2.0",
 "method" : "UI.OnKeyboardInput",
 "params" :
 {
  "event" : "ENTRY_CANCELLED"
 }
}
```

OnLanguageChange

Type

Notification

Sender

HMI

Purpose

Inform SDL that the UI display language has changed.

Notification

PARAMETERS

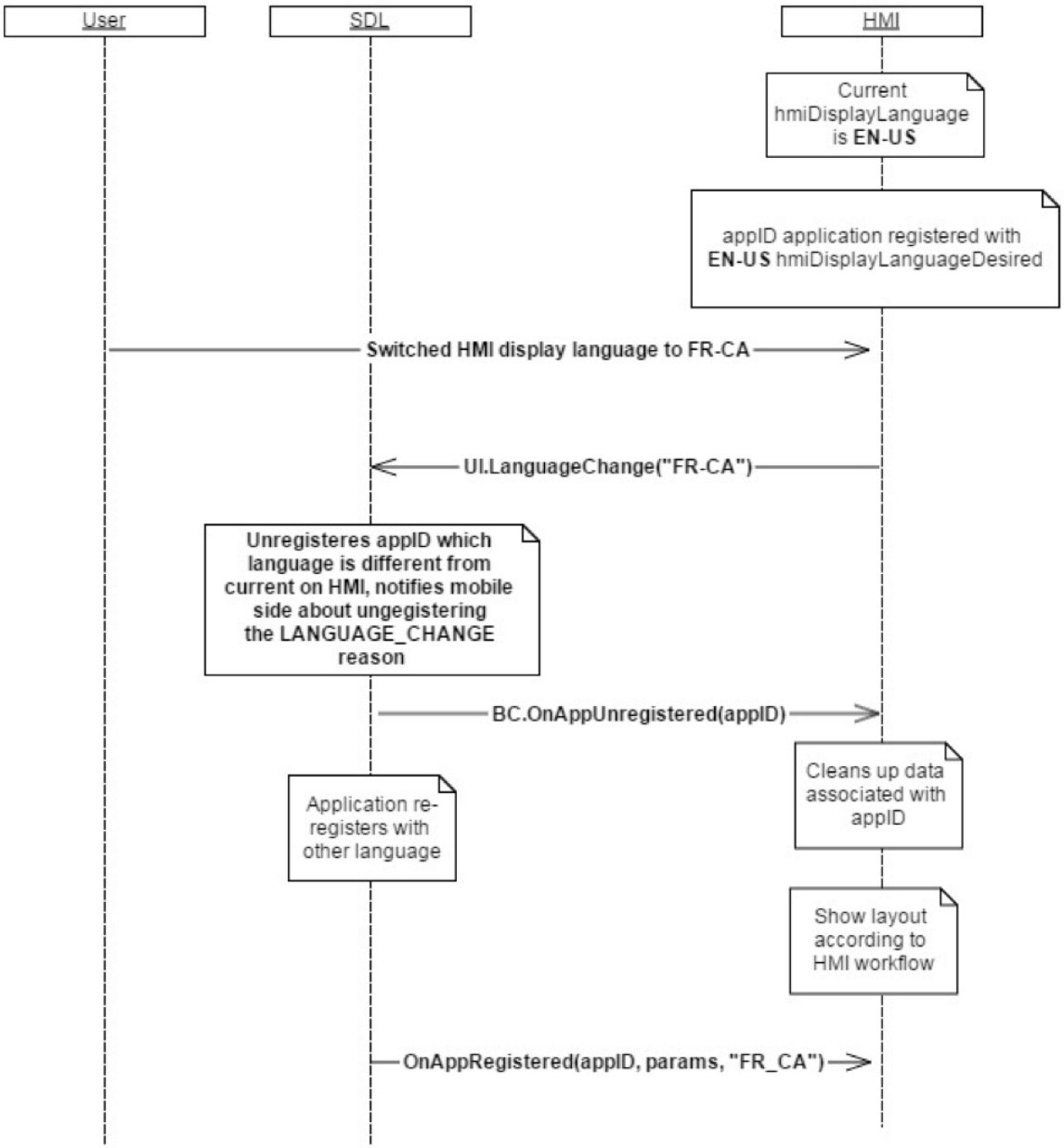
NAME	TYPE	MANDATORY	ADDITIONAL
language	Common.Language	true	

Sequence Diagrams

SEQUENCE DIAGRAM

OnLanguageChange

View Diagram



JSON EXAMPLE NOTIFICATION

```
{
 "jsonrpc" : "2.0",
 "method" : "UI.OnLanguageChange",
 "params" :
 {
  "language" : "FR-CA"
 }
}
```

OnRecordStart

Notification

PARAMETERS

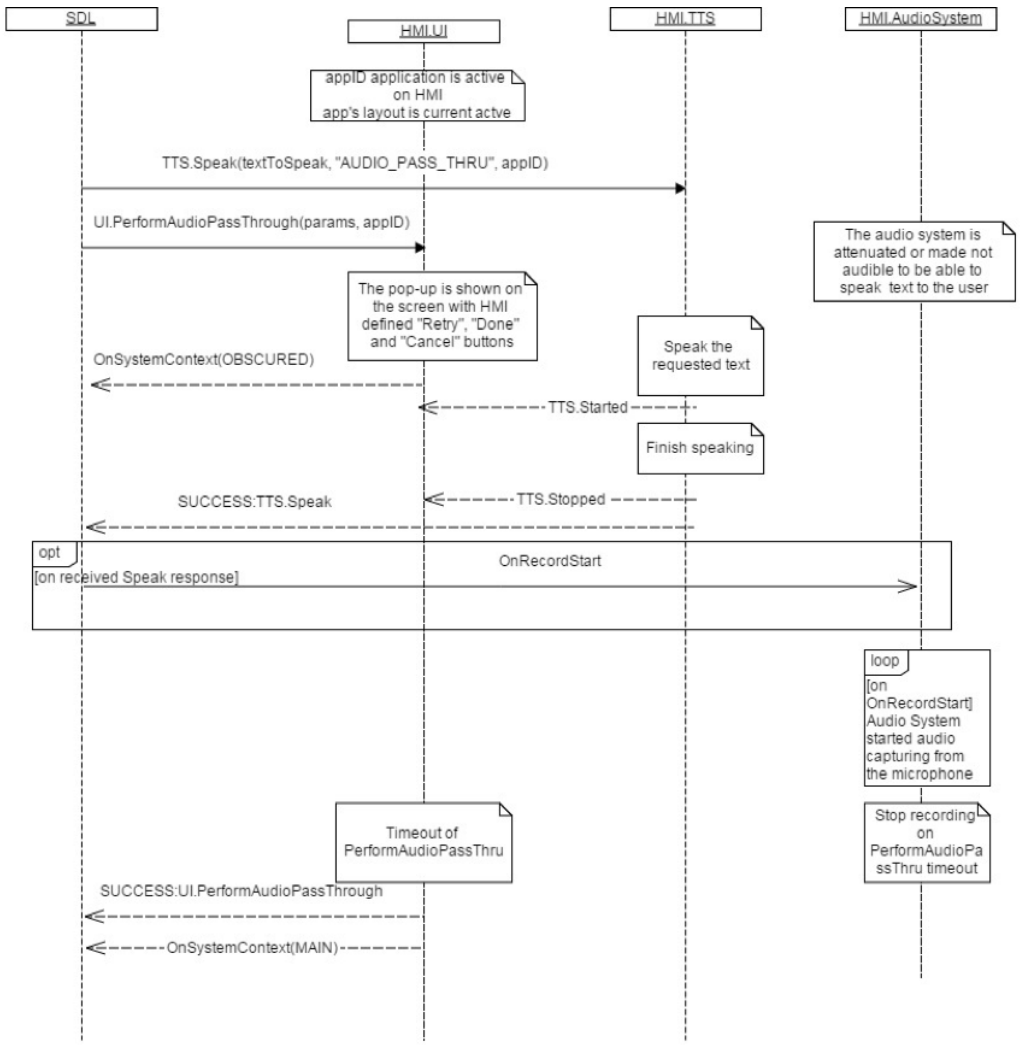
NAME	TYPE	MANDATORY	ADDITIONAL
appID	Integer	true	

Sequence Diagrams

SEQUENCE DIAGRAM

OnRecordStart with TTS.Speak

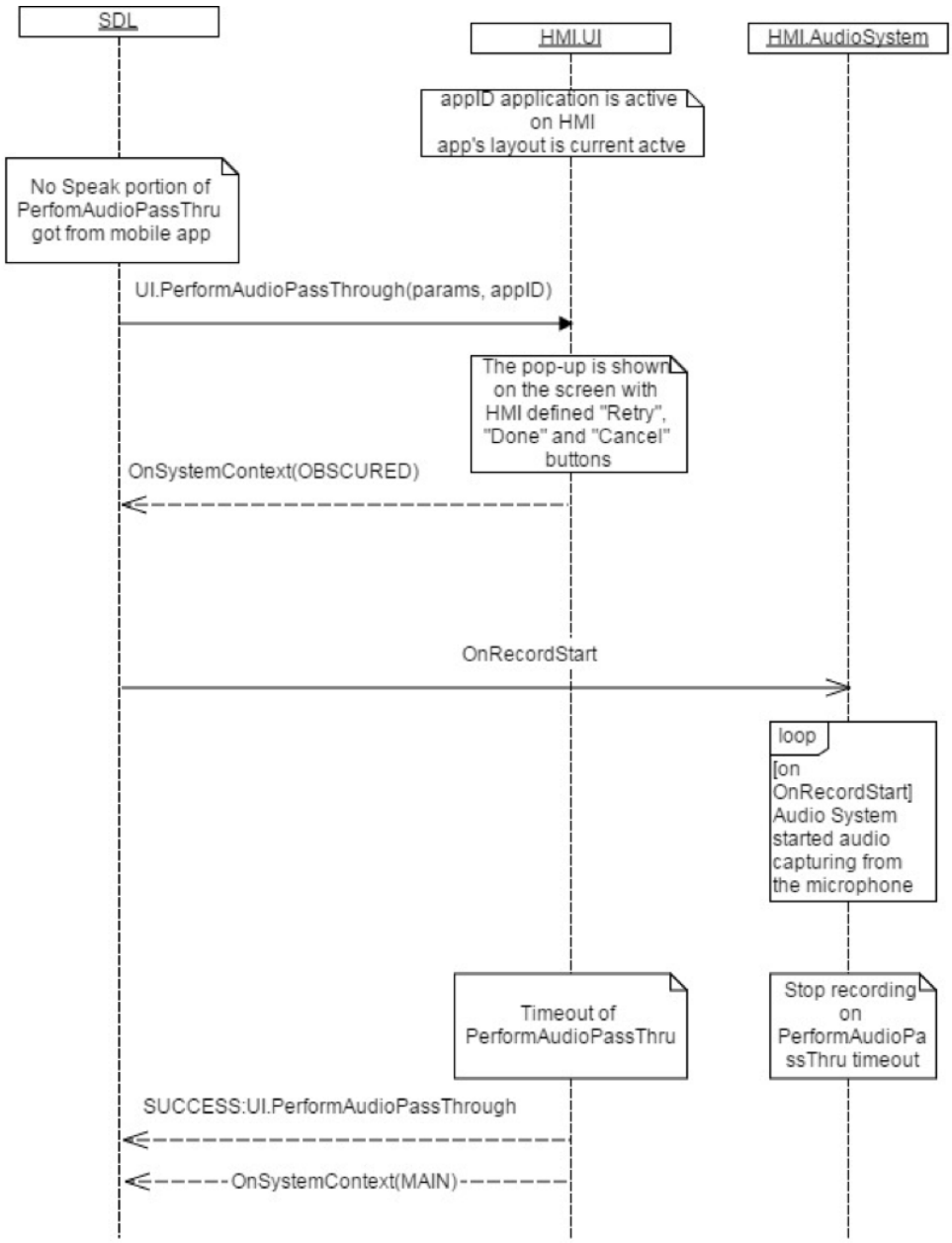
View Diagram



SEQUENCE DIAGRAM

OnRecordStart without TTS.Speak

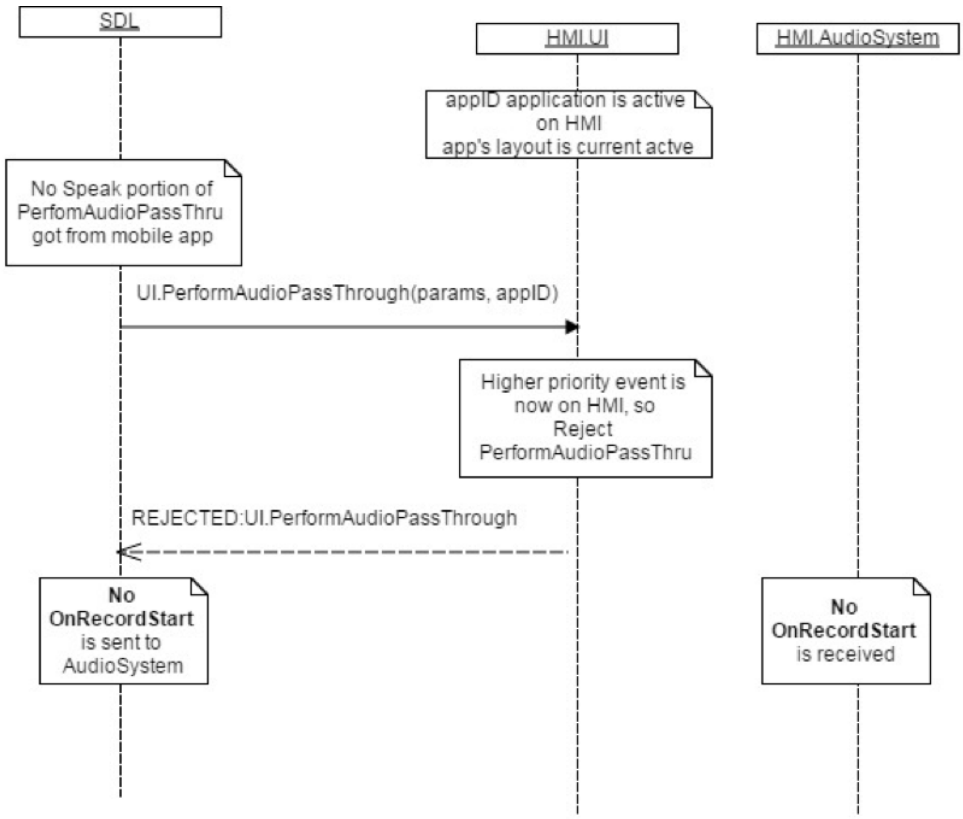
View Diagram



SEQUENCE DIAGRAM

OnRecordStart not sent if UI.PerformAudioPassThru rejected

View Diagram



JSON EXAMPLE NOTIFICATION

```
{
 "jsonrpc" : "2.0",
 "method" : "UI.OnRecordStart",
 "params" :
 {
  "appID" : 65537
 }
}
```

OnResetTimeout

Type Notification
Sender HMI
Purpose Inform SDL that a timeout has been reset via user action for some transient UI element.

Notification

PARAMETERS

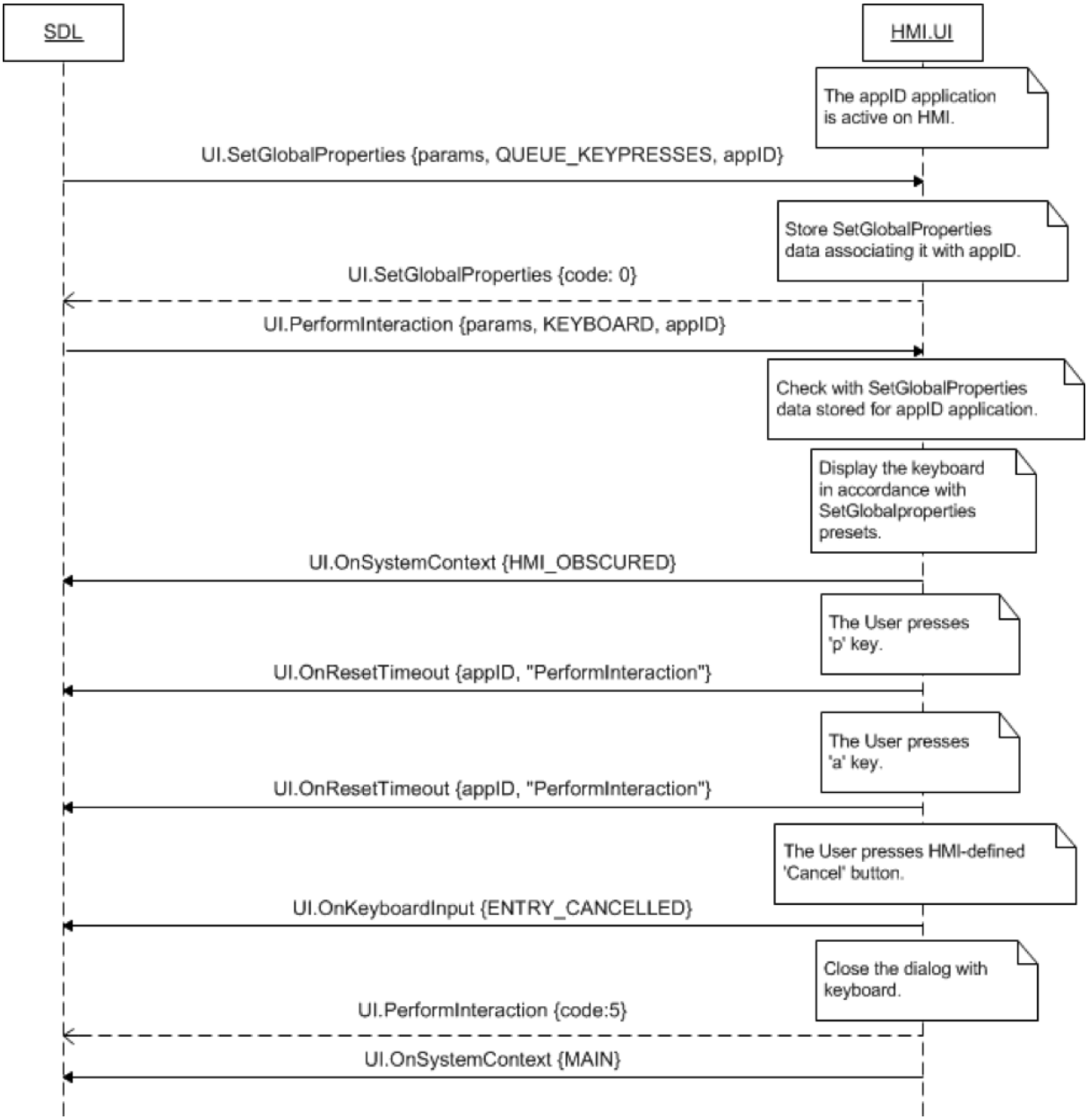
NAME	TYPE	MANDATORY	ADDITIONAL
appID	Integer	true	
methodName	String	true	

Sequence Diagrams

SEQUENCE DIAGRAM

OnResetTimeout during PerformInteraction

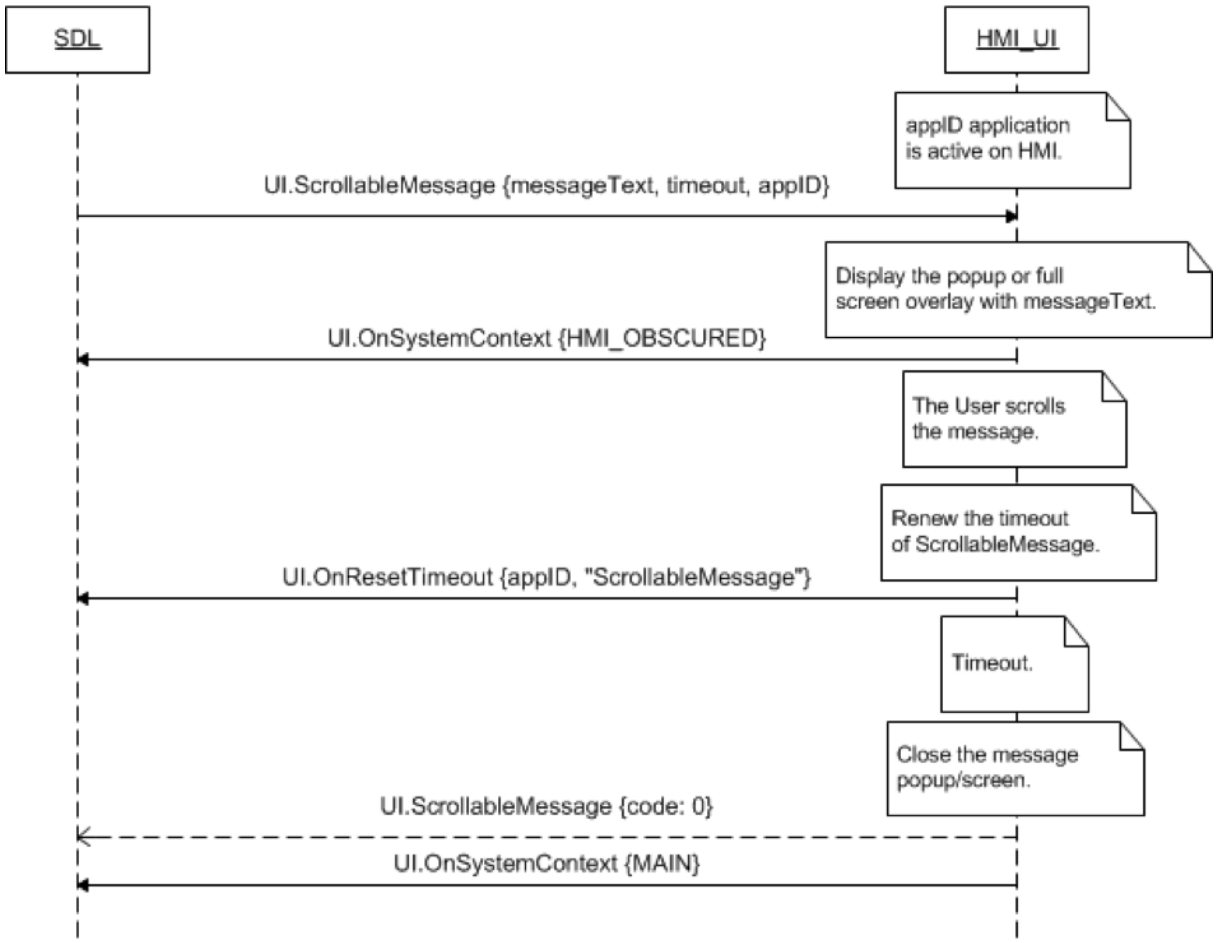
View Diagram



SEQUENCE DIAGRAM

OnResetTimeout during ScrollableMessage

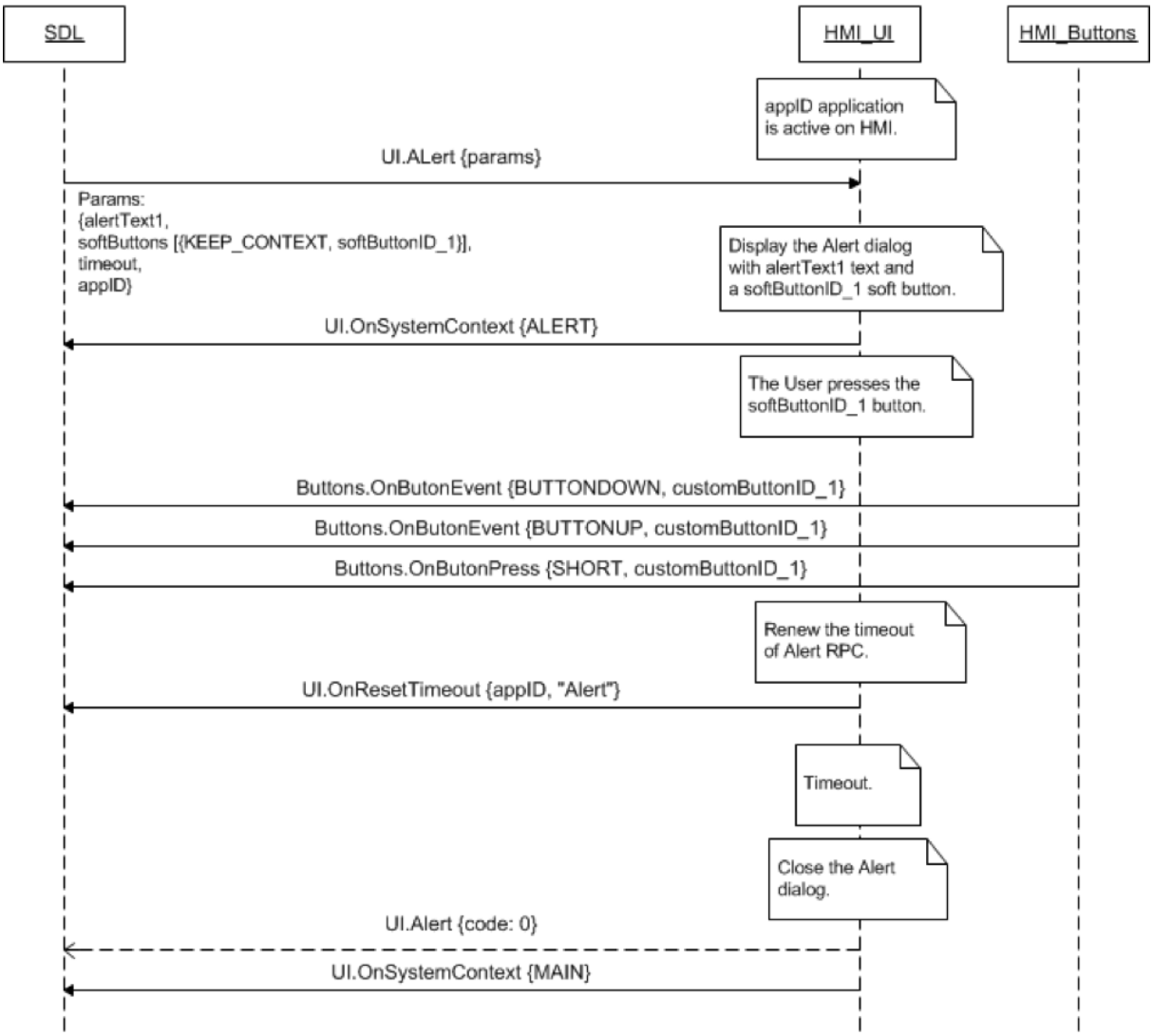
View Diagram



SEQUENCE DIAGRAM

OnResetTimeout KEEP_CONTEXT during alert

View Diagram



JSON EXAMPLE NOTIFICATION

```
{
  "jsonrpc" : "2.0",
  "method" : "UI.OnResetTimeout",
}
```

OnSystemContext

Type

Notification

Sender

HMI

Purpose

Inform SDL about the current context of app interaction.

Notification

PARAMETERS

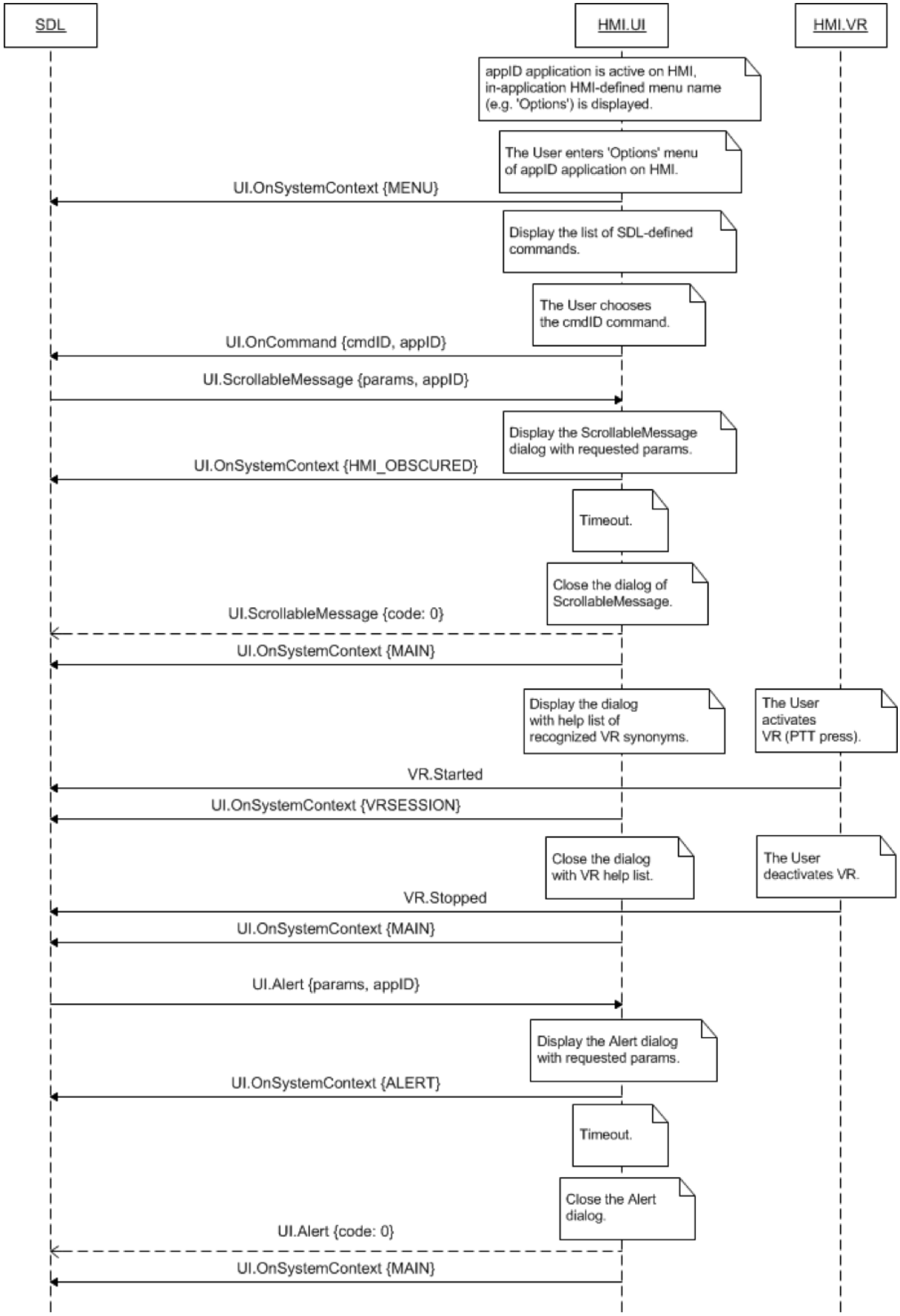
NAME	TYPE	MANDATORY	ADDITIONAL
systemContext	Common.System Context	true	
appID	Integer	false	

Sequence Diagrams

SEQUENCE DIAGRAM

OnSystemContext for different HMI States

View Diagram


SDL
HMI.UI
HMI.VR
appID application is active on HMI, in-application HMI-defined menu name (e.g. 'Options') is displayed.
The User enters 'Options' menu of appID application on HMI.
UI.OnSystemContext {MENU}
Display the list of SDL-defined commands.
The User chooses the cmdID command.
UI.OnCommand {cmdID, appID}
UI.ScrollableMessage {params, appID}
Display the ScrollableMessage dialog with requested params.
UI.OnSystemContext {HMI_OBSCURED}
Timeout.
Close the dialog of ScrollableMessage.
UI.ScrollableMessage {code: 0}
UI.OnSystemContext {MAIN}
Display the dialog with help list of recognized VR synonyms.
The User activates VR (PTT press).
VR.Started
UI.OnSystemContext {VRSESSION}
Close the dialog with VR help list.
The User deactivates VR.
VR.Stopped
UI.OnSystemContext {MAIN}
UI.Alert {params, appID}
Display the Alert dialog with requested params.
UI.OnSystemContext {ALERT}
Timeout.
Close the Alert dialog.
UI.Alert {code: 0}
UI.OnSystemContext {MAIN}

JSON EXAMPLE NOTIFICATION

```
{
 "jsonrpc" : "2.0",
 "method" : "UI.OnSystemContext",
 "params" :
 {
  "systemContext" : "VRSESSION"
 }
}
```

OnTouchEvent

Type Notification
Sender HMI
Purpose

Inform SDL that a touch event has occurred in the HMI.

Notification

PARAMETERS

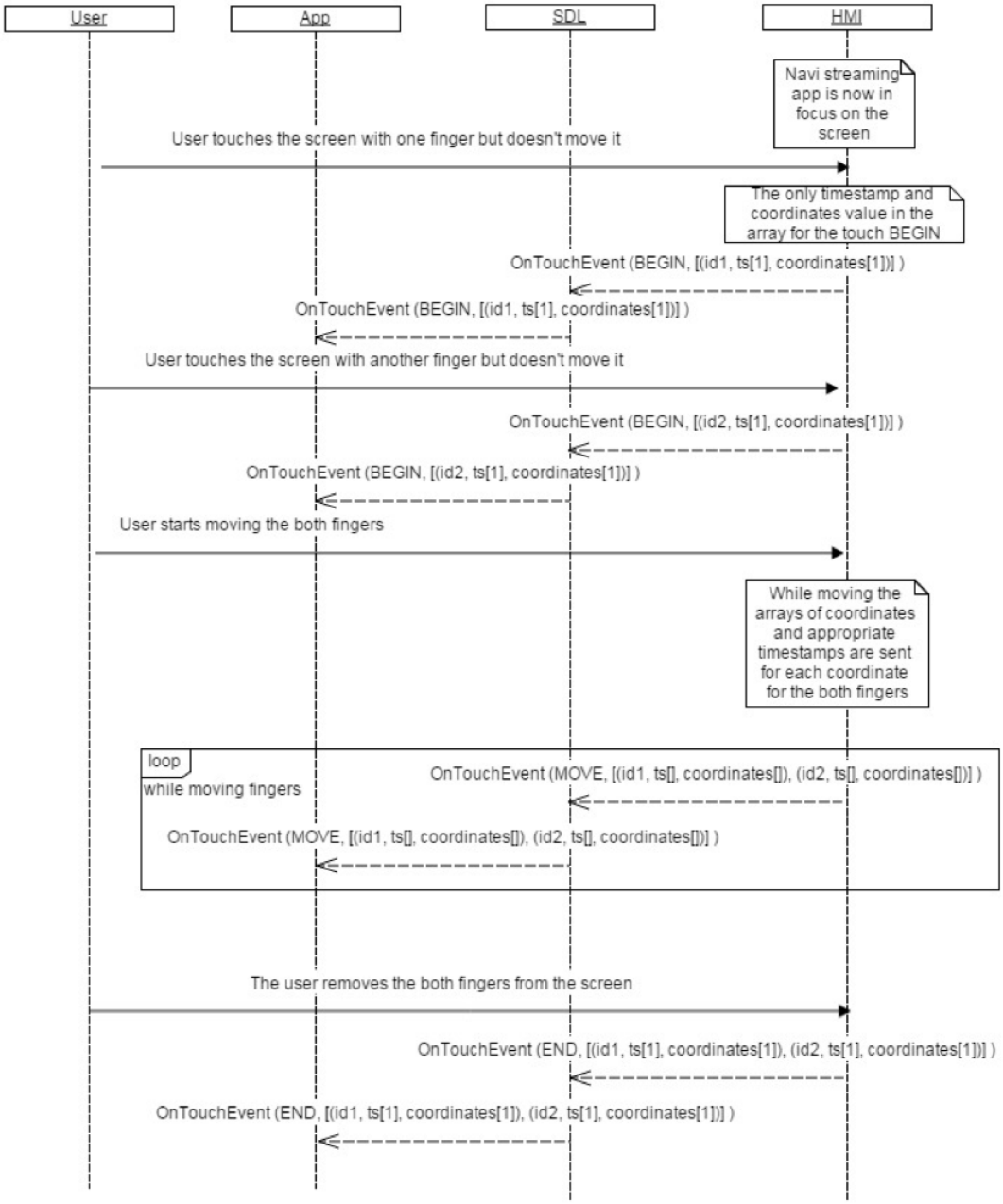
NAME	TYPE	MANDATORY	ADDITIONAL
type	Common.TouchType	true	
event	Common.TouchEvent	true	array: true minsize: 1 maxsize: 10

Sequence Diagrams

SEQUENCE DIAGRAM

OnTouchEvent moving two fingers stop together

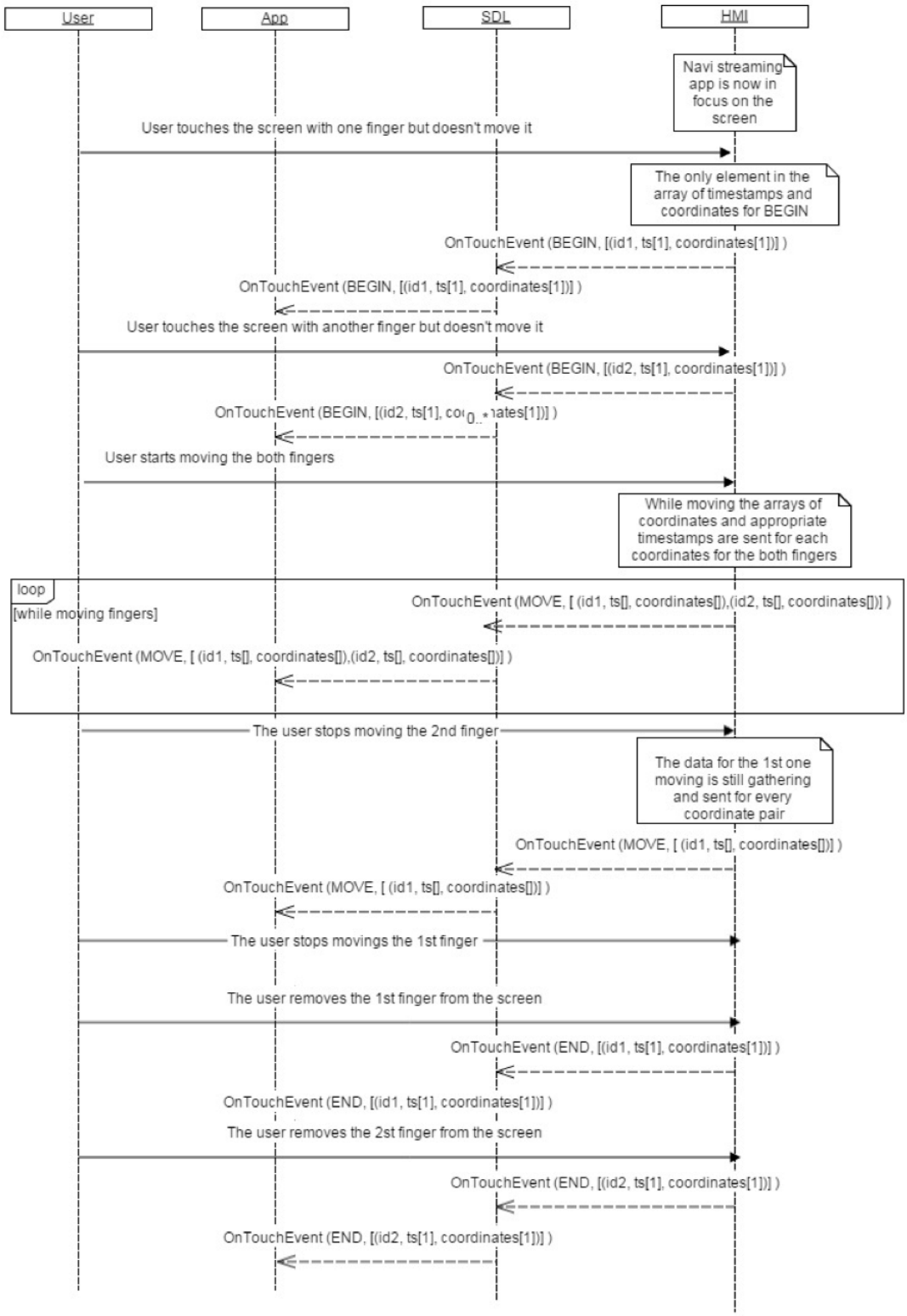
View Diagram



SEQUENCE DIAGRAM

OnTouchEvent moving two fingers one stops first

View Diagram


User
App
SDL
HMI
Navi streaming app is now in focus on the screen
User touches the screen with one finger but doesn't move it
The only element in the array of timestamps and coordinates for BEGIN
OnTouchEvent (BEGIN, [(id1, ts[1], coordinates[1])])
OnTouchEvent (BEGIN, [(id1, ts[1], coordinates[1])])
User touches the screen with another finger but doesn't move it
OnTouchEvent (BEGIN, [(id2, ts[1], coordinates[1])])
OnTouchEvent (BEGIN, [(id2, ts[1], coordinates[1])])
User starts moving the both fingers
While moving the arrays of coordinates and appropriate timestamps are sent for each coordinates for the both fingers
loop
[while moving fingers]
OnTouchEvent (MOVE, [(id1, ts[], coordinates[]),(id2, ts[], coordinates[])])
OnTouchEvent (MOVE, [(id1, ts[], coordinates[]),(id2, ts[], coordinates[])])
The user stops moving the 2nd finger
The data for the 1st one moving is still gathering and sent for every coordinate pair
OnTouchEvent (MOVE, [(id1, ts[], coordinates[])])
OnTouchEvent (MOVE, [(id1, ts[], coordinates[])])
The user stops movings the 1st finger
The user removes the 1st finger from the screen
OnTouchEvent (END, [(id1, ts[1], coordinates[1])])
OnTouchEvent (END, [(id1, ts[1], coordinates[1])])
The user removes the 2st finger from the screen
OnTouchEvent (END, [(id2, ts[1], coordinates[1])])
OnTouchEvent (END, [(id2, ts[1], coordinates[1])])

JSON EXAMPLE NOTIFICATION

```
{
 "jsonrpc" : "2.0",
 "method" : "UI.OnTouchEvent",
 "params" :
 {
  "type" : START,
   " event" :[
        {
          "id":0,
          "ts":[49013],
          "c":[{"x":323,"y":259}],
         }
       ]
 }
}
```

PerformAudioPassThru

Type Function
Sender SDL
Purpose

Start audio capturing and sending PCM data to SDL

REQUEST

PARAMETERS

NAME	TYPE	MANDATORY	ADDITIONAL
appID	Integer	true	
audioPassThruDisplayTexts	Common.TextFieldStruct	true	array: true minsize: 0 maxsize: 2
maxDuration	Integer	true	minvalue: 1 maxvalue: 1000000
muteAudio	Boolean	true	

RESPONSE

PARAMETERS

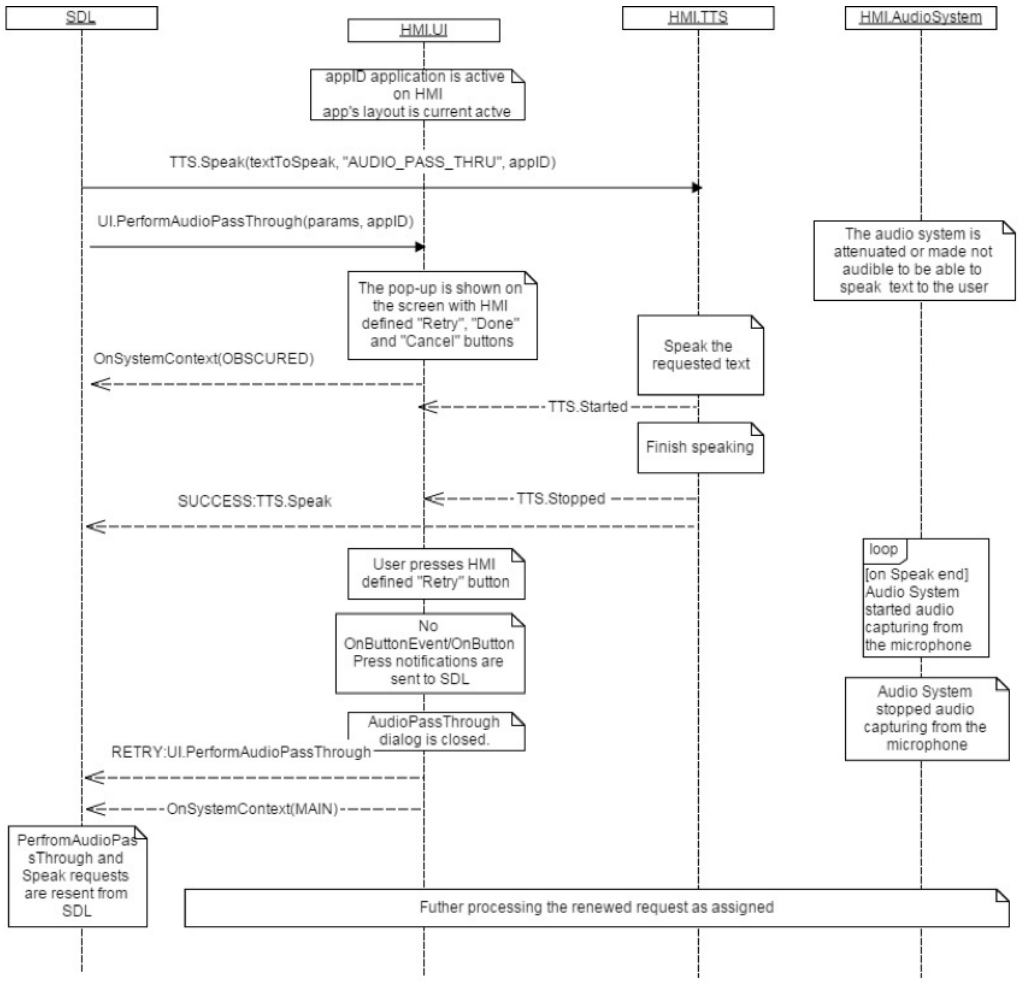
This RPC has no additional parameter requirements

Sequence Diagrams

SEQUENCE DIAGRAM

PerformAudioPassThru requested with TTS.Speak

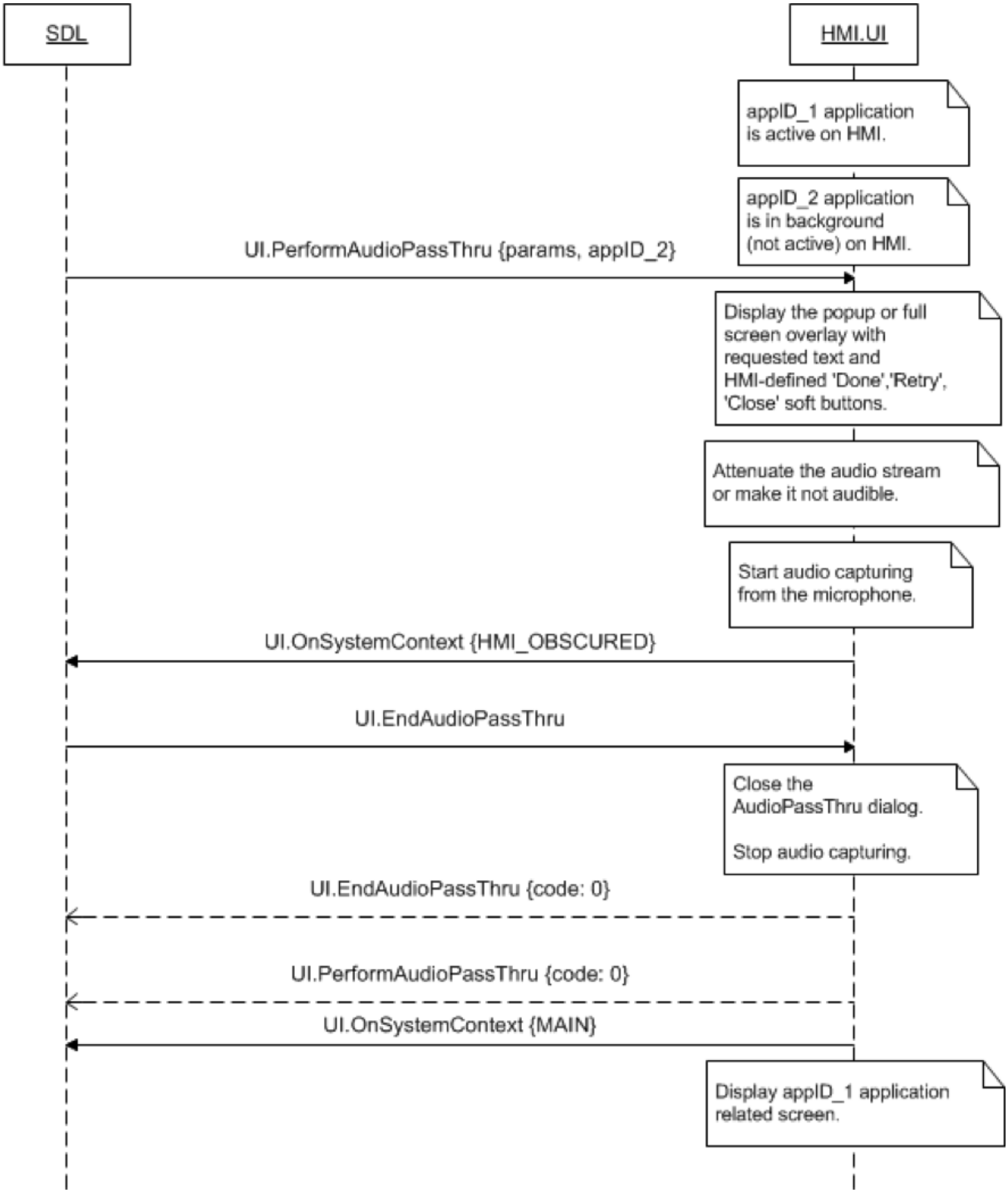
View Diagram



SEQUENCE DIAGRAM

PerformAudioPassThru with EndAudioPassThru

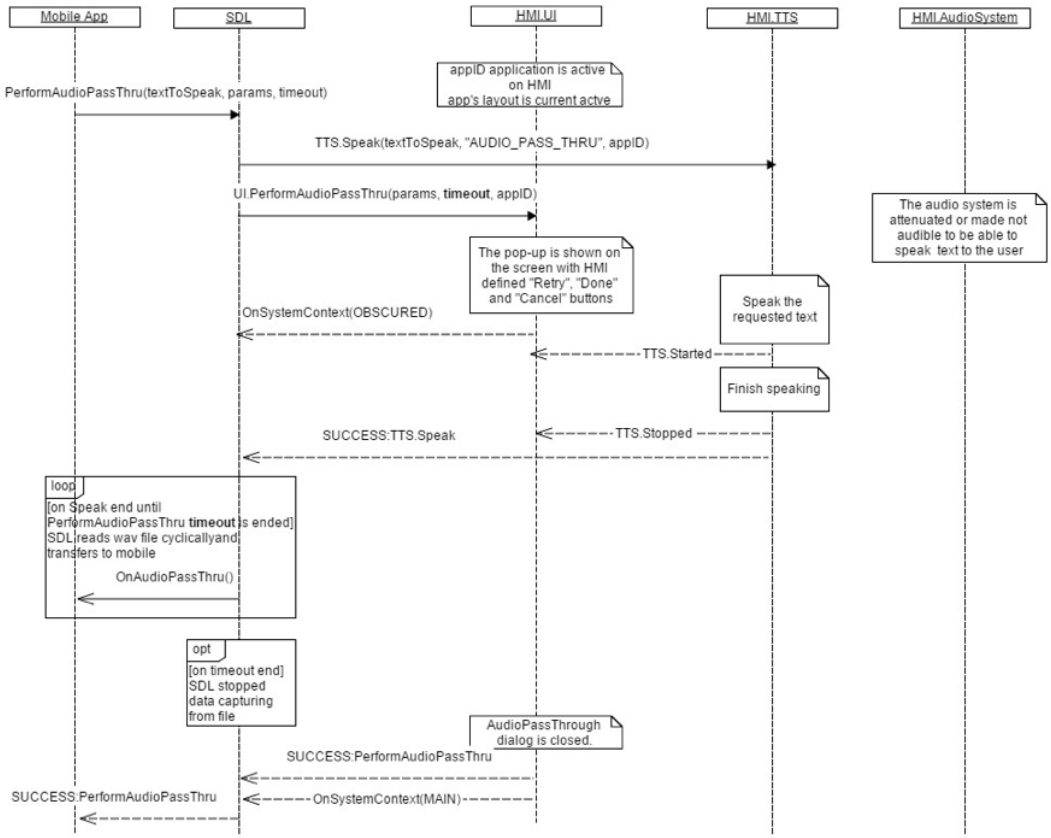
View Diagram



SEQUENCE DIAGRAM

PerformAudioPassThru not supported

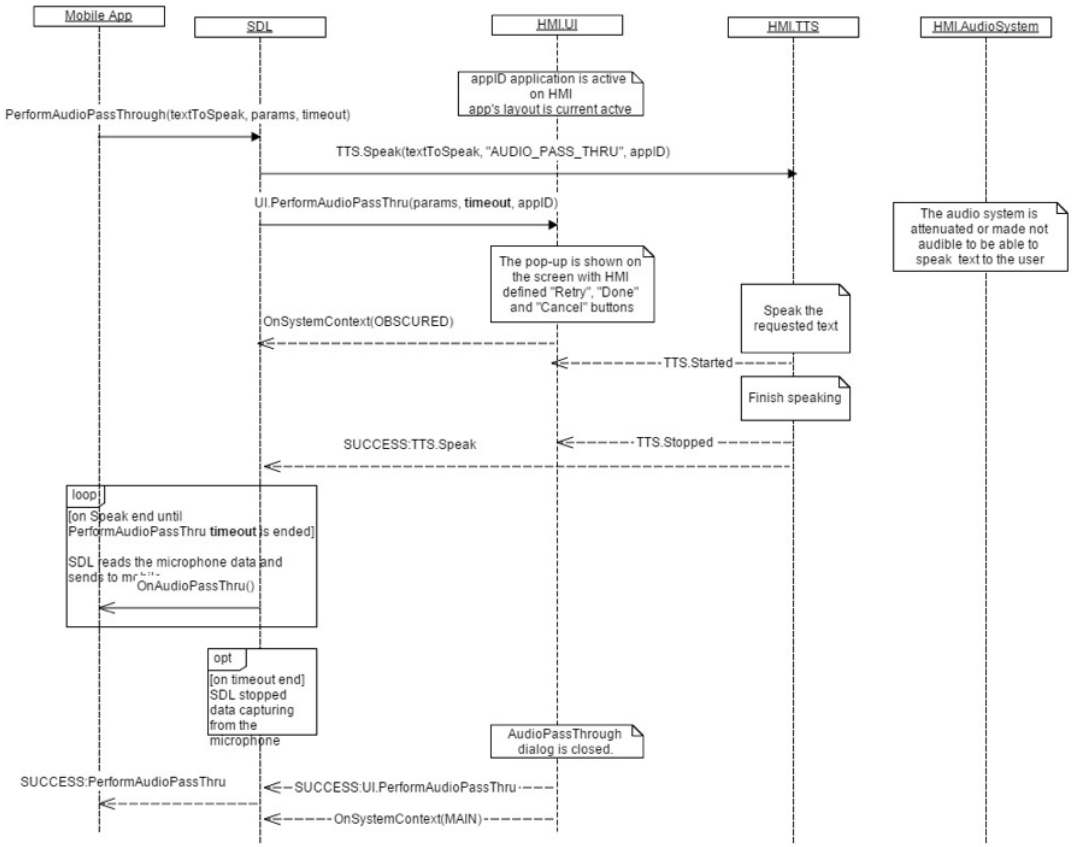
View Diagram



SEQUENCE DIAGRAM

PerformAudioPassThru from vehicle microphone

View Diagram



Example Request

```
{
 "id" : 79,
 "jsonrpc" : "2.0",
 "method" : "UI.PerformAudioPassThru",
 "params" :
 {
  "audioPassThruDisplayTexts" :
  {
      "fieldName" :  audioPassThruDisplayText1,
   "fieldText" : "The audio capturing is in progress"
  },
  "maxDuration" : 10000,
 }
}
```

Example Response

```
{
 "id" : 79,
 "jsonrpc" : "2.0",
 "result" :
 {
  "code" : 0,
  "method" : "UI.PerformAudioPassThru"
 }
}
```

Example Error

```
{
 "id" : 79,
 "jsonrpc" : "2.0",
 "error" :
 {
  "code" : 7,
  "message" : "The user interrupted the RPC and indicated to start
over",
  "data" :
  {
   "method" : "UI.PerformAudioPassThru"
  }
 }
}
```

PerformInteraction

Type Function
Sender SDL
Purpose Perform a UI Interaction with the User.

A request sent by SDL to display a list of choices to the user.

MUST

1. Display the choices in `choiceSet` to the user reasonably according to the `interactionLayout`
2. Wait until the user responds or the request times out to respond to SDL
3. Include the `choiceID` of the chosen option in the response to SDL
4. If the interaction times out, respond with a success and no `choiceID` parameter to SDL

NOTE

A UI.PerformInteraction with a timeout value of `0` should not be timed out immediately. Instead, it should use some default timeout value predetermined by the HMI.

REQUEST

PARAMETERS

NAME	TYPE	MANDATORY	ADDITIONAL
initialText	Common.TextFieldStruct	false	
choiceSet	Common.Choice	false	array: true minsize: 1 maxsize: 100
vrHelpTitle	String	false	maxlength: 500
vrHelp	Common.VrHelpItem	false	array: true minsize: 1 maxsize: 100
timeout	Integer	true	minvalue: 5000 maxvalue: 100000
interactionLayout	Common.LayoutMode	false	
appID	Integer	true	

RESPONSE

PARAMETERS

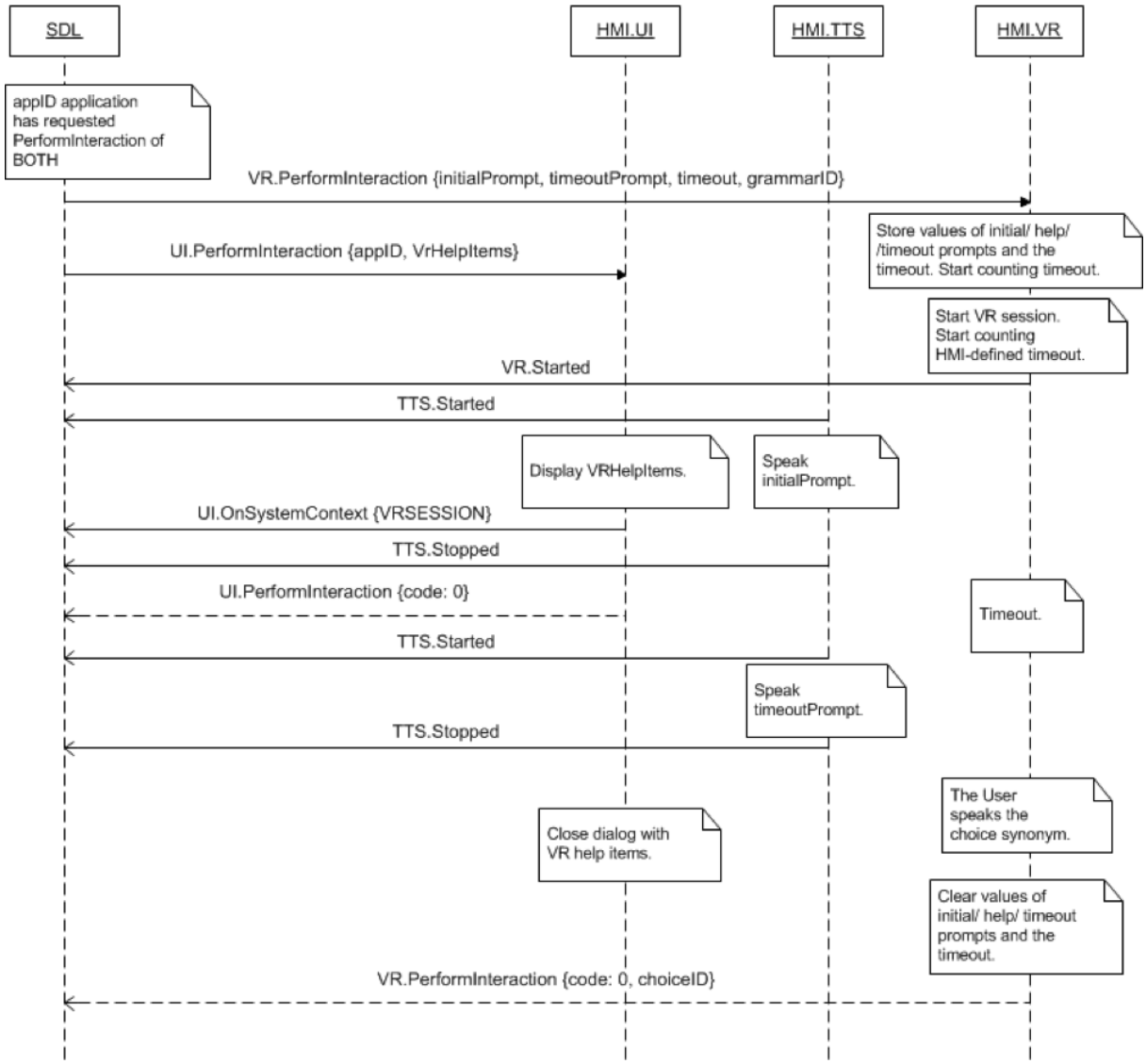
NAME	TYPE	MANDATORY	ADDITIONAL
choiceID	Integer	false	minvalue: 0 maxvalue: 2000000000
manualTextEntry	String	false	minlength: 0 maxlength: 500

Sequence Diagrams

SEQUENCE DIAGRAM

PerformInteraction Successful with VR Only

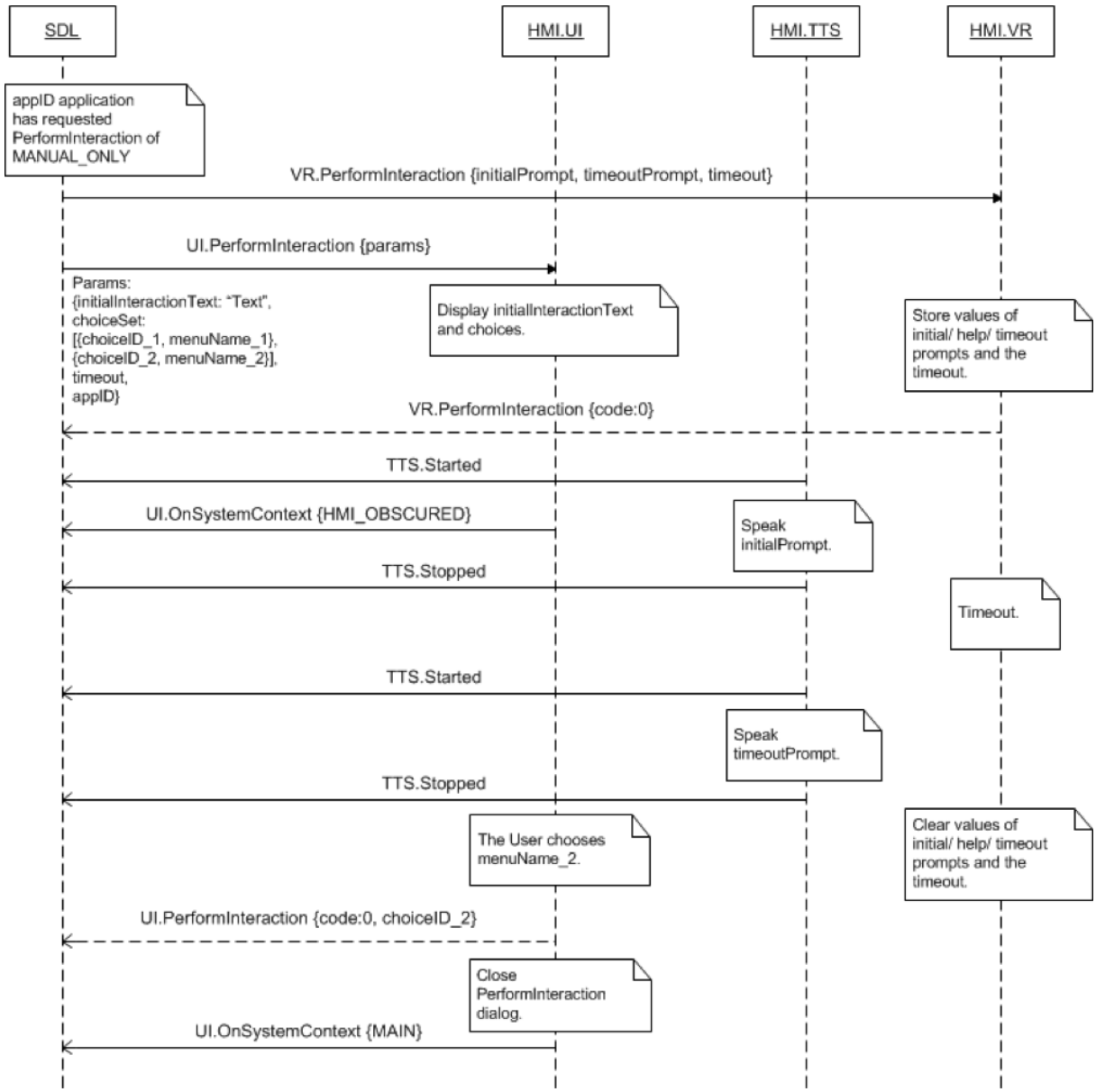
View Diagram



SEQUENCE DIAGRAM

PerformInteraction Successful with Manual Only

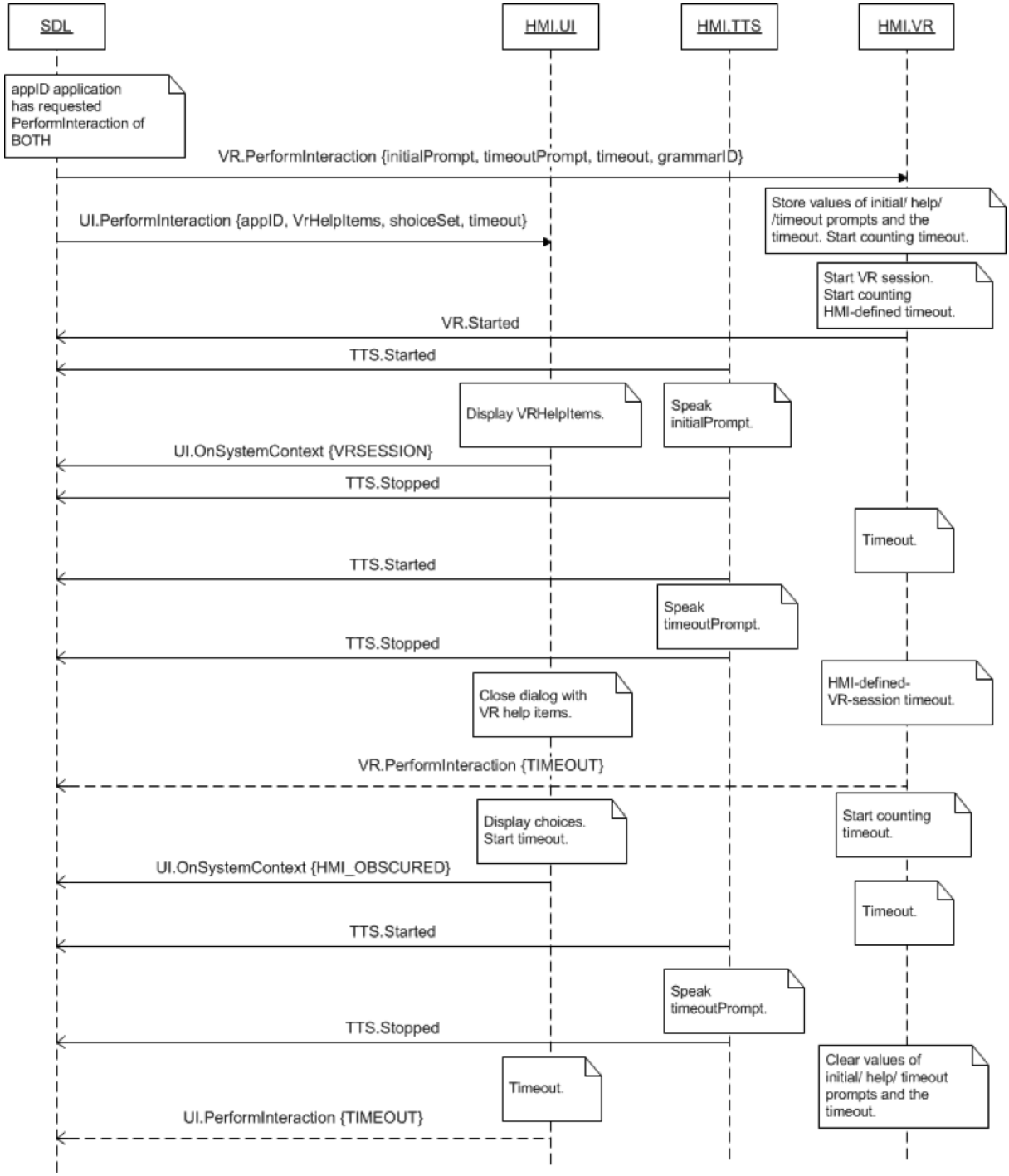
View Diagram



SEQUENCE DIAGRAM

PerformInteraction Timeout with Both

View Diagram


SDL
HMI.UI
HMI.TTS
HMI.VR
appID application has requested PerformInteraction of BOTH
VR.PerformInteraction {initialPrompt, timeoutPrompt, timeout, grammarID}
Store values of initial/ help/ /timeout prompts and the timeout. Start counting timeout.
UI.PerformInteraction {appID, VrHelpItems, shoiceSet, timeout}
Start VR session. Start counting HMI-defined timeout.
VR.Started
TTS.Started
Display VRHelpItems.
Speak initialPrompt.
UI.OnSystemContext {VRSESSION}
TTS.Stopped
Timeout.
TTS.Started
Speak timeoutPrompt.
TTS.Stopped
Close dialog with VR help items.
HMI-defined-VR-session timeout.
VR.PerformInteraction {TIMEOUT}
Display choices. Start timeout.
Start counting timeout.
UI.OnSystemContext {HMI_OBSCURED}
Timeout.
TTS.Started
Speak timeoutPrompt.
TTS.Stopped
Timeout.
Clear values of initial/ help/ timeout prompts and the timeout.
UI.PerformInteraction {TIMEOUT}

Example Request

```
{
 "id" : 79,
 "jsonrpc" : "2.0",
 "method" : "UI.PerformInteraction",
 "params" :
 {
  "initialText" :
  {
      "fieldName" : initialInteractionText,
      "fieldText" : "Choose the station:"
  },

  "choiceSet" :
  [
   {
     "choiceID" : 2415,
     "menuName" : "Sky.FM"
   },

   {
     "choiceID" : 2416,
     "menuName" : "Paradise"
   },

   {
     "choiceID" : 2417,
     "menuName" : "100 XR"
   }
  ],

  "vrHelp" :
  [
     {
      "text" : "Sky FM",
      "image" :
       [
         "value" : "tmp/SDL/app/Pandora/icon_5410.jpg",
         "imageType" : DYNAMIC
       ],

      "position" : 1
     },

     {
      "text" : "Paradise",
      "image" :
       [
         "value" : "tmp/SDL/app/Pandora/icon_5423.jpeg",
         "imageType" : DYNAMIC
```

```
        ],
        "position" : 2
       },

       {
        "text" : "100 XR",
        "image" :
        [
          "value" : "tmp/SDL/app/Pandora/icon_5465.jpeg",
          "imageType" : DYNAMIC
        ],
        "position" : 3
       }
    ],

    "timeout" : 15000,
    "appID" : 6493
  }
}
```

Example Response

```
{
  "id" : 79,
  "jsonrpc" : "2.0",
  "result" :
  {
    "choiceID" : 2416
    "code" : 0,
    "method" : "UI.PerformInteraction"
  }
}
```

Example Error


```
{
 "id" : 79,
 "jsonrpc" : "2.0",
 "error" :
 {
  "code" : 10,
  "message" : "Overlay reached the maximum timeout and closed",
  "data" :
  {
   "method" : "UI.PerformInteraction"
  }
 }
}
```


ScrollableMessage

Type Function
Sender SDL
Purpose Display a dialog that may contain new lines and text which is scrollable by the user.

REQUEST

PARAMETERS

NAME	TYPE	MANDATORY	ADDITIONAL
messageText	Common.TextFieldStruct	true	
timeout	Integer	true	minvalue: 0 maxvalue: 65535
softButtons	Common.SoftButton	false	array: true minsize: 0 maxsize: 8
appID	Integer	true	

RESPONSE

PARAMETERS

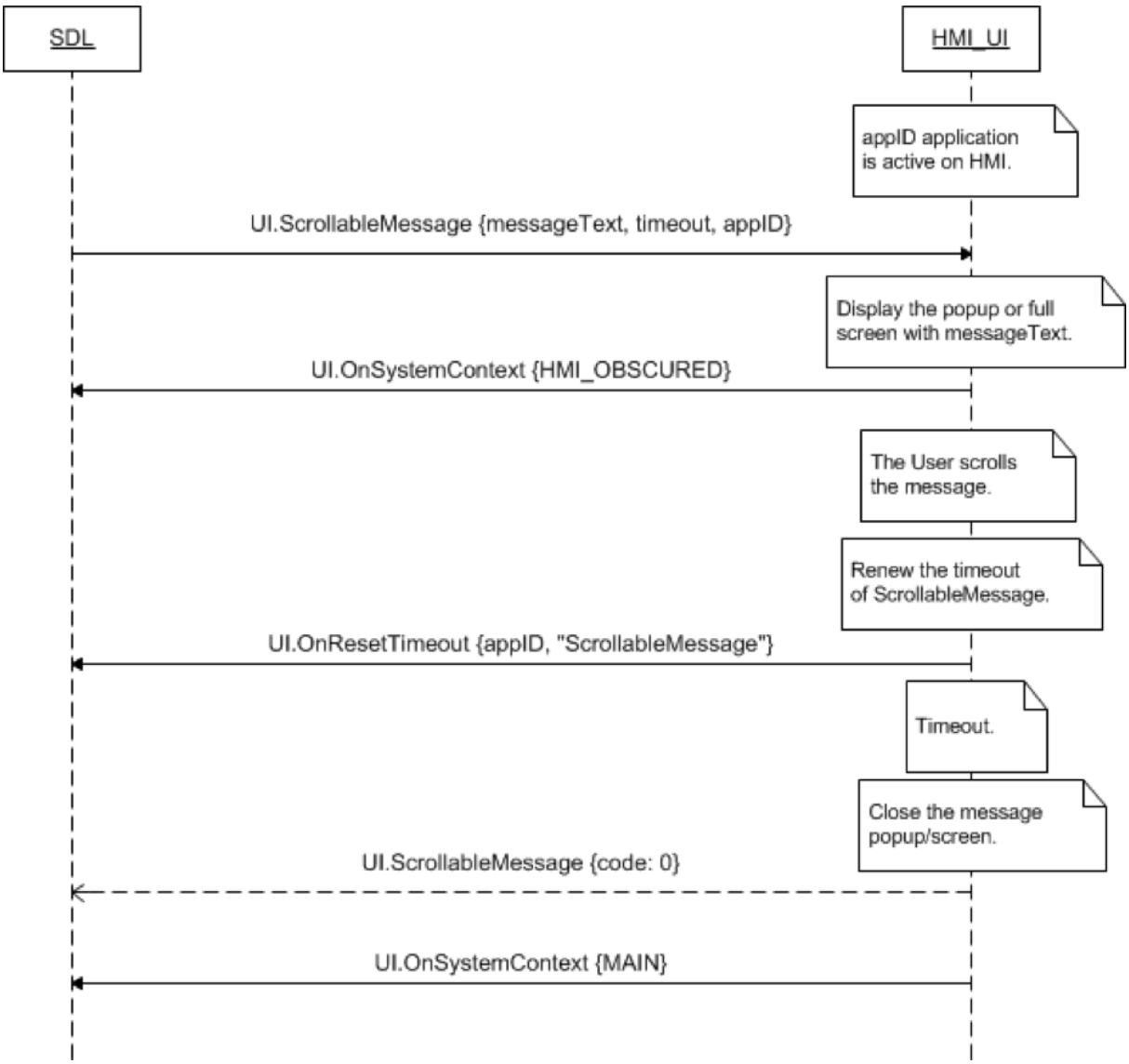
This RPC has no additional parameter requirements

Sequence Diagrams

SEQUENCE DIAGRAM

ScrollableMessage scrolled and closed by timeout

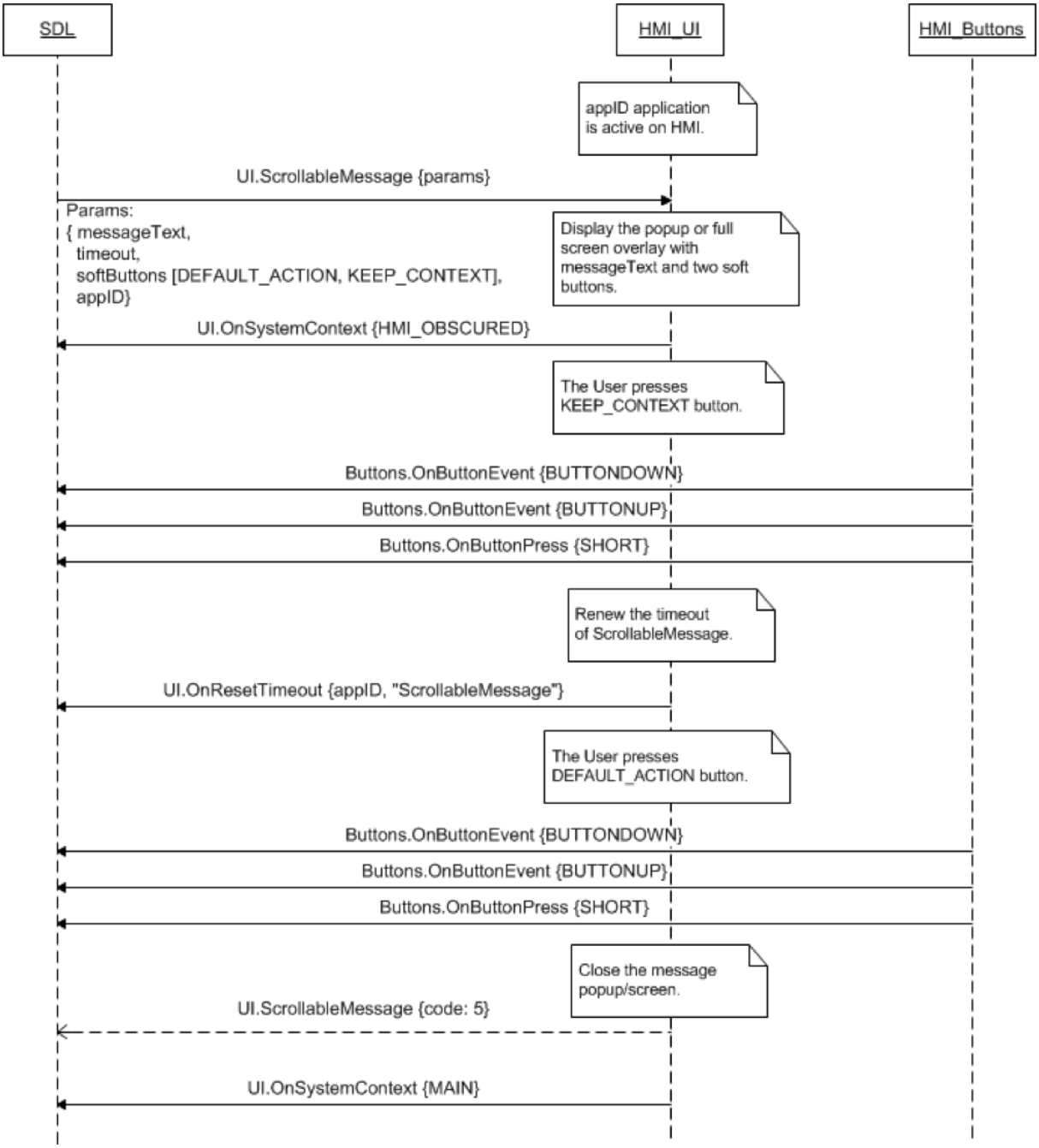
View Diagram



SEQUENCE DIAGRAM

ScrollableMessage with soft buttons pressed by user

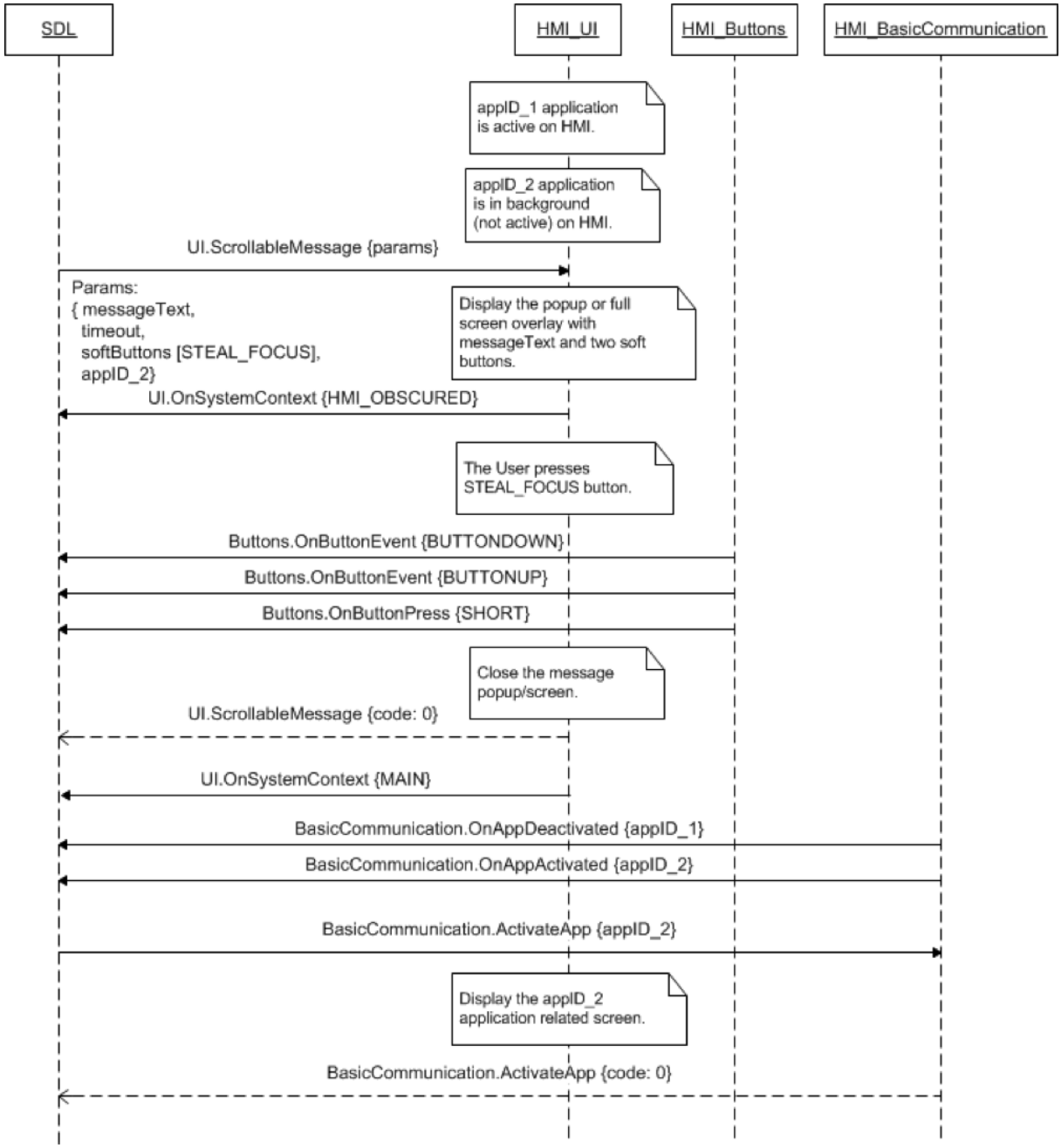
View Diagram



SEQUENCE DIAGRAM

ScrollableMessage with STEAL_FOCUS button for background application

View Diagram


SDL
HMI_UI
HMI_Buttons
HMI_BasicCommunication
appID_1 application is active on HMI.
appID_2 application is in background (not active) on HMI.
UI.ScrollableMessage {params}
Params:
{ messageText,
timeout,
softButtons [STEAL_FOCUS],
appID_2}
Display the popup or full screen overlay with messageText and two soft buttons.
UI.OnSystemContext {HMI_OBSCURED}
The User presses STEAL_FOCUS button.
Buttons.OnButtonEvent {BUTTONDOWN}
Buttons.OnButtonEvent {BUTTONUP}
Buttons.OnButtonPress {SHORT}
Close the message popup/screen.
UI.ScrollableMessage {code: 0}
UI.OnSystemContext {MAIN}
BasicCommunication.OnAppDeactivated {appID_1}
BasicCommunication.OnAppActivated {appID_2}
BasicCommunication.ActivateApp {appID_2}
Display the appID_2 application related screen.
BasicCommunication.ActivateApp {code: 0}

Example Request

```
{
  "id" : 138,
  "jsonrpc" : "2.0",
  "method" : "UI.ScrollableMessage",
  "params" :
  {
    "messageText" :
    {
        "fieldName" :  scrollableMessageBody,
     "fieldText" : "Create a Station
         Enter an artist, song or composer in the Search box in the top left corner. We'll create a radio station featuring that music and more like it. You can also create a new station from the song or artist currently playing by hovering over the album artwork, clicking the white up-arrow and selecting New Station—you can choose From Song or From Artist."
    },
    "timeout" : 10000,
    "softButtons" :
    [
      {
        "type" :  TEXT,
        "text" : "Leave onscreen",
        "softButtonID" : 15,
        "systemAction" :  KEEP_CONTEXT
      },
      {
        "type" :  TEXT,
        "text" : "Cancel",
        "softButtonID" : 16,
        "systemAction" :  STEAL_FOCUS
      }
    ],

    "appID" : 6527
  }
}
```

Example Response


```
{
 "id" : 138,
 "jsonrpc" : "2.0",
 "result" :
 {
  "code" : 0,
  "method" : "UI.ScrollableMessage"
 }
}
```


Example Error


```
{
 "id" : 138,
 "jsonrpc" : "2.0",
 "error" :
 {
  "code" : 12,
  "message" : "The string data is too long",
  "data" :
  {
   "method" : "UI.ScrollableMessage"
  }
 }
}
```

SendHapticData

Type Function
Sender SDL
Purpose Communicate areas of focus in a video streaming application to the HMI

UI.SendHapticData represents a request for the HMI to keep track of several application-defined areas within a video stream for focus purposes. The HMI should highlight these areas with a rectangle when they are brought into focus on the HMI.

MUST

The HMI must store the provided HapticRect data and utilize it whenever the users switches focus to part of the application that sent the message.

MUST

The HMI must replace all stored HapticRect data every time a new SendHapticData is received for a given application. When an empty hapticRectData array is received, the HMI must clear any existing HapticRect data for that application.

MUST

The HMI must send a pair UI.OnTouchEvent (one BEGIN and one END) with a TouchCoord corresponding to the center of the HapticRect when one of these elements is selected by the user.

REQUEST

PARAMETERS

NAME	TYPE	MANDATORY	ADDITIONAL
hapticRectData	Common.HapticRect	true	array: true minSize: 0 maxSize: 1000
appID	Integer	true	

RESPONSE

PARAMETERS

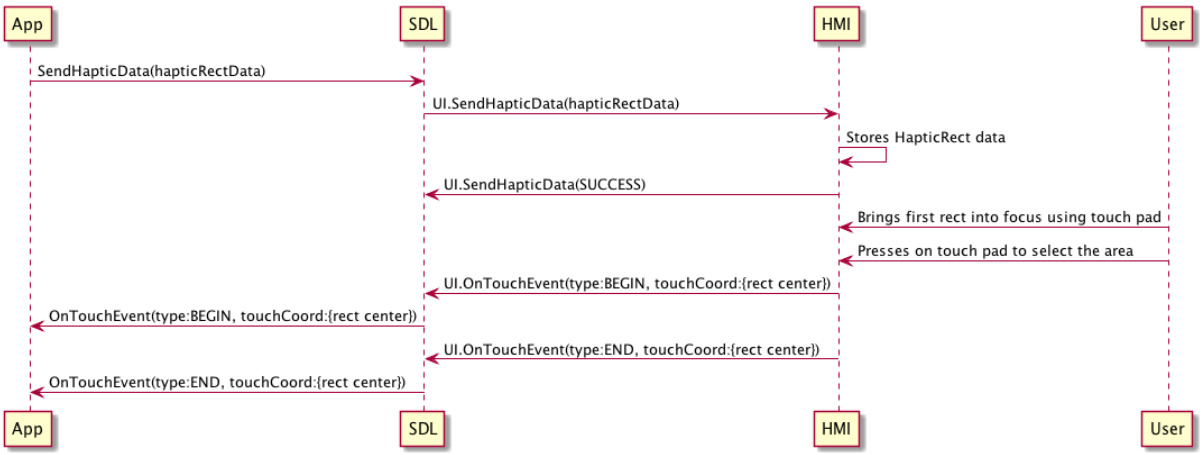
This RPC has no additional parameter requirements

Sequence Diagrams

SEQUENCE DIAGRAM

SendHapticData basic flow with user selection

View Diagram



Example Request

```
{
 "id" : 70,
 "jsonrpc" : "2.0",
 "method" : "UI.SendHapticData",
 "params" :
 {
  "hapticRectData" :
  [
   {
    "id" : 234,
    "rect" :
    {
     "x" : 567.2,
     "y" : 342.4,
     "width" : 200.0,
     "height": 150.0
    }
   }
  ],
  "appID" : 65464
 }
}
```

Example Response

```
{
 "id" : 70,
 "jsonrpc" : "2.0",
 "result" :
 {
  "code" : 0,
  "method" : "UI.SendHapticData"
 }
}
```

Example Error

```
{
 "id" : 70,
 "jsonrpc" : "2.0",
 "error" :
 {
  "code" : 2,
  "message" : "The HMI does not support the use of haptic data",
  "data" :
  {
   "UI.SendHapticData"
  }
 }
}
```

SetAppIcon

Type Function
Sender SDL
Purpose Display the application's specified icon in the HMI's application list.

REQUEST

MUST

1. Keep track and store the the name and file referencing the app icon each time a successful `SetAppIcon` request occurs from the mobile app.
2. Display the requested icon together with the application name in the HMI list of registered applications:
    - Right away if the the list of registered applications is currently displayed
    - When the list of registered applications is displayed upon User's request on HMI
3. Display default app icon if the mobile application had not previously set an icon successfully and/or the file does not exist on HMI.

PARAMETERS

NAME	TYPE	MANDATORY	ADDITIONAL
syncFileName	Common.Image	true	
appID	Integer	true	

RESPONSE

PARAMETERS

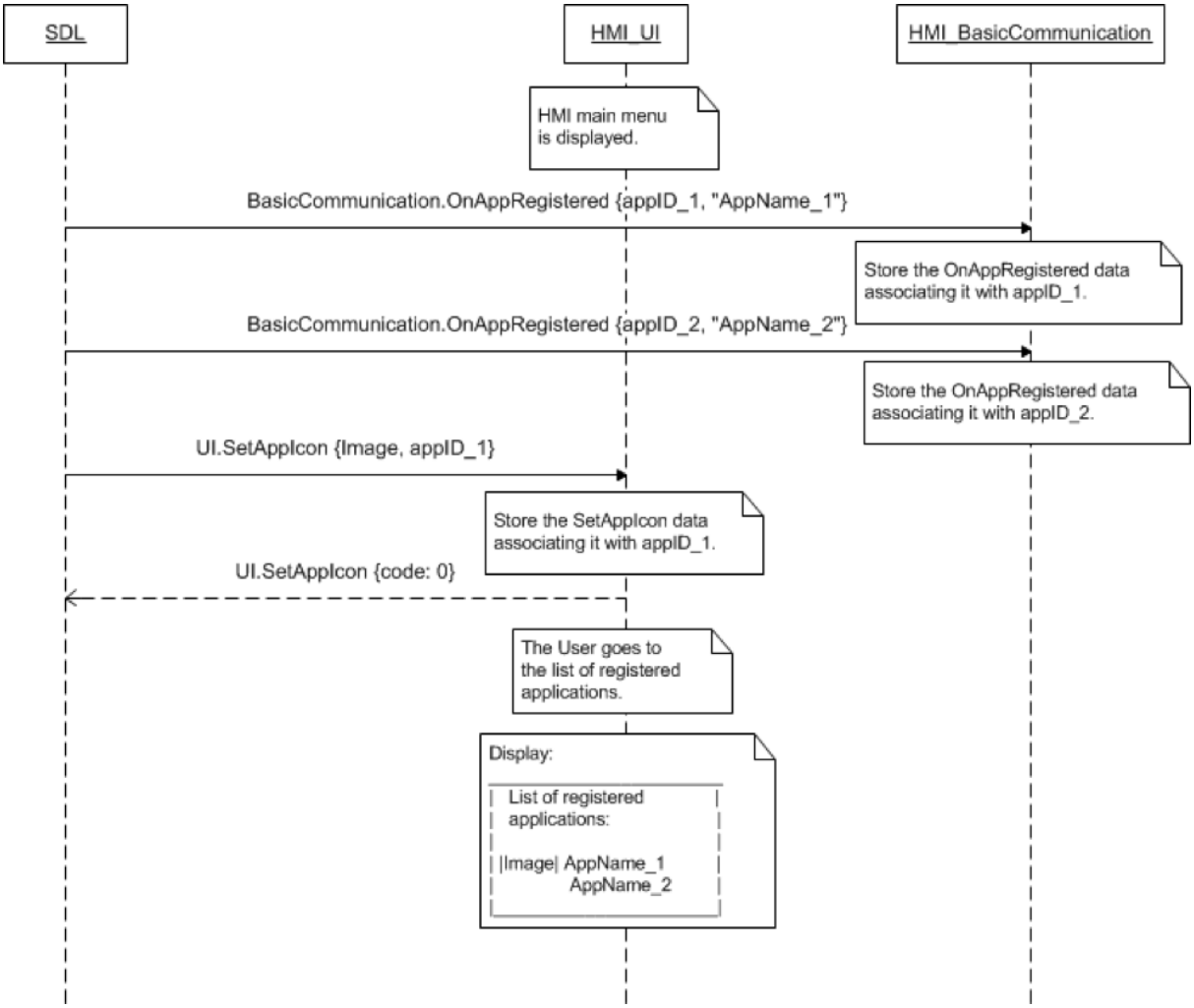
This RPC has no additional parameter requirements

Sequence Diagrams

SEQUENCE DIAGRAM

SetAppIcon

View Diagram



Example Request

```
{
 "id" : 88,
 "jsonrpc" : "2.0",
 "method" : "UI.SetAppIcon",
 "params" :
 {
  "syncFileName" :
  {
      "value" : "tmp/SDL/app/Best_Media/12345.jpg",
      "imageType" : "DYNAMIC"
  },
  "appID" : 65146
 }
}
```

Example Response


```
{
 "id" : 88,
 "jsonrpc" : "2.0",
 "result" :
 {
  "code" : 0,
 "method" : "UI.SetAppIcon"
 }
}
```


Example Error


```
{
 "id" : 88,
 "jsonrpc" : "2.0",
 "error" :
 {
  "code" : 2,
  "message" : "Unsupported resource",
  "data" :
  {
   "method" : "UI.SetAppIcon"
  }
 }
}
```

SetDisplayLayout

Type

Function

Sender

SDL

Purpose

Set the display template for the specified application's persistent display.

REQUEST

PARAMETERS

NAME	TYPE	MANDATORY	ADDITIONAL
displayLayout	String	true	maxlength: 500
appID	Integer	true	
dayColorScheme	Common.TemplateColorScheme	false	
nightColorScheme	Common.TemplateColorScheme	false	

RESPONSE

PARAMETERS

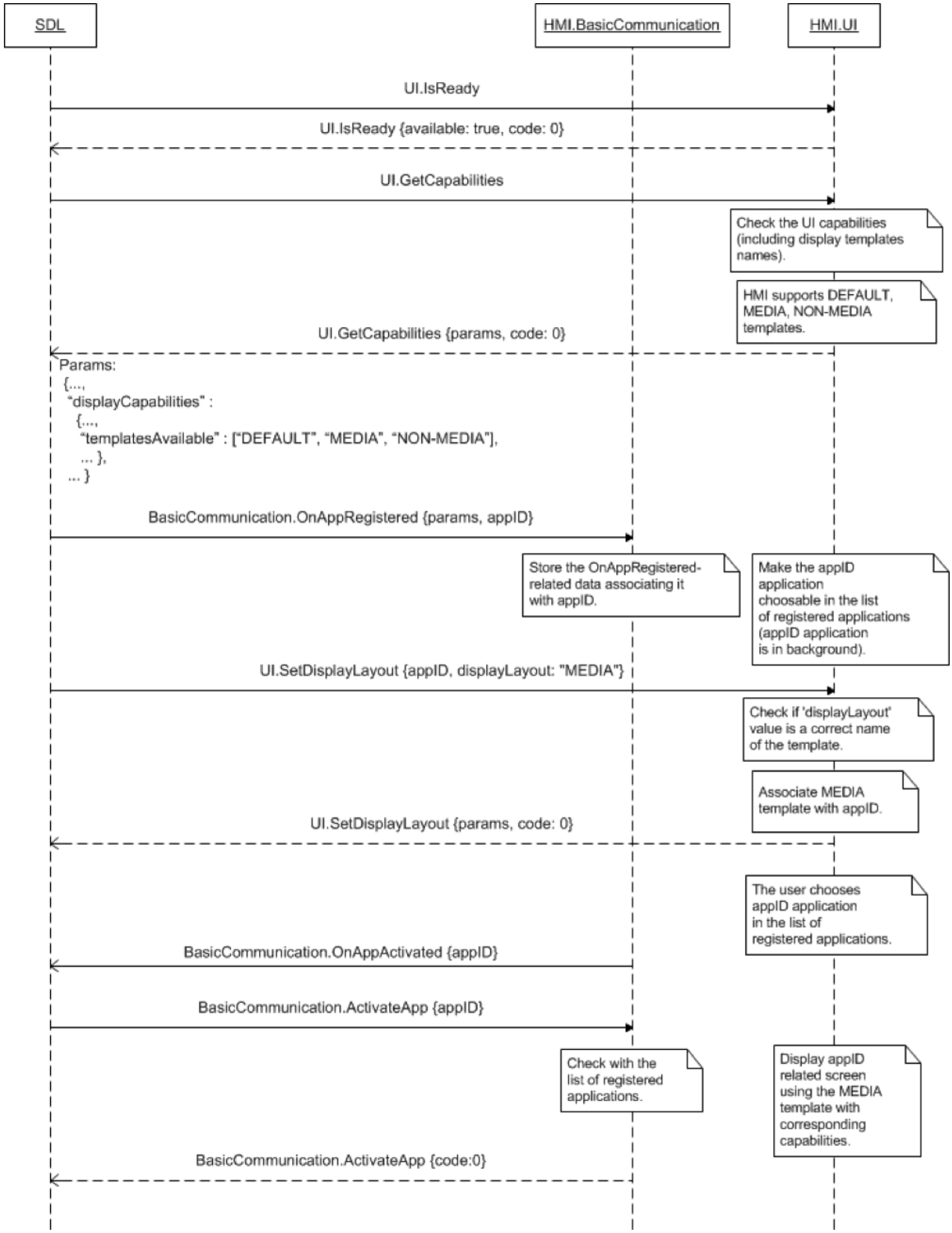
NAME	TYPE	MANDATORY	ADDITIONAL
displayCapabilities	Common.DisplayCapabilities	false	
buttonCapabilities	Common.ButtonCapabilities	false	array: true minsize: 1 maxsize: 100
softButtonCapabilities	Common.SoftButtonCapabilities	false	array: true minsize: 1 maxsize: 100
presetBankCapabilities	Common.PresetBankCapabilities	false	

Sequence Diagrams

SEQUENCE DIAGRAM

SetDisplayLayout Successful with UI.GetCapabilities

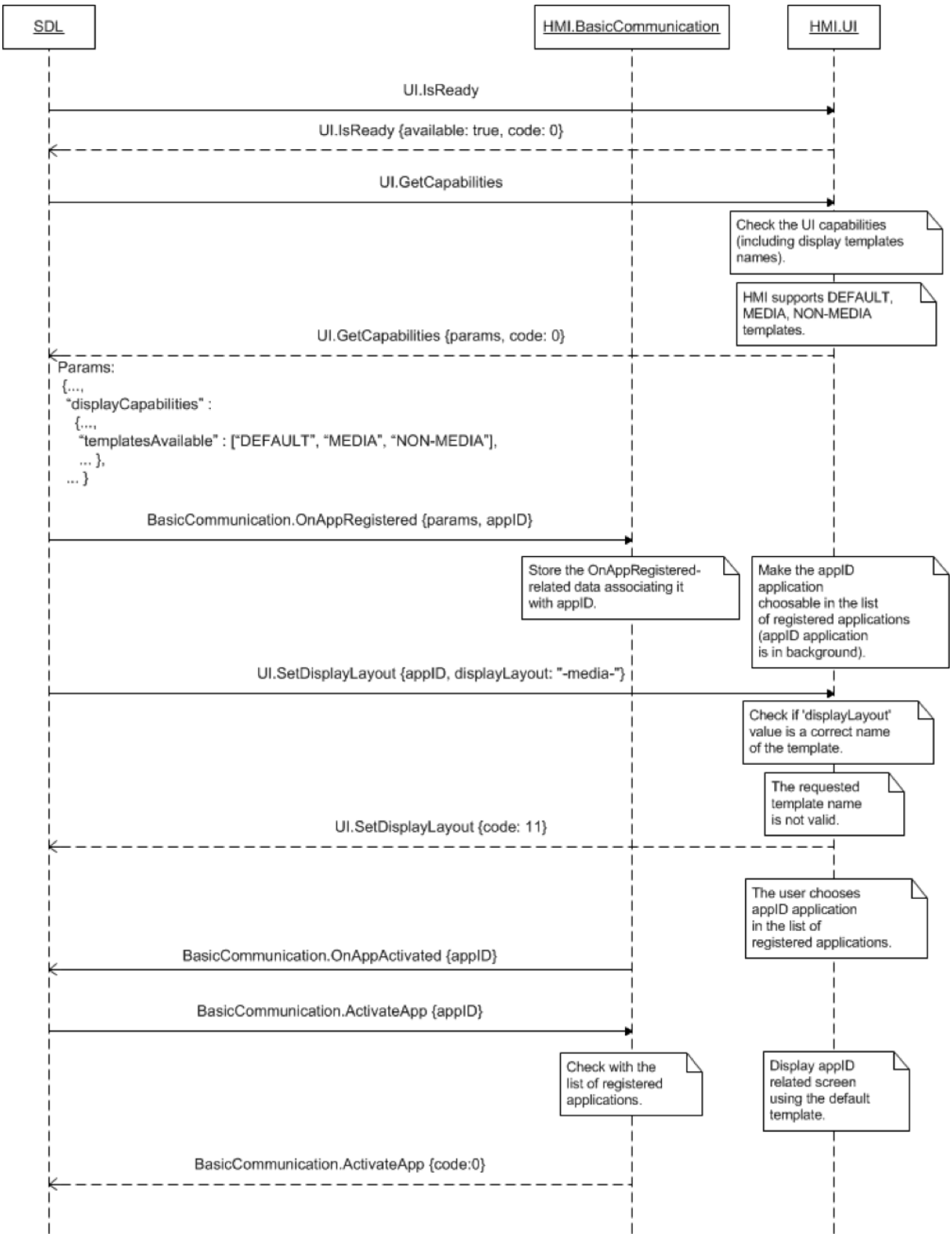
View Diagram



SEQUENCE DIAGRAM

SetDisplayLayout Invalid Data with UI.GetCapabilities

View Diagram



Example Request

```
{
 "id" : 47,
 "jsonrpc" : "2.0",
 "method" : "UI.SetDisplayLayout",
 "params" :
 {
  "displayLayout" : "NON-MEDIA",
  "appID" : 65638
 }
}
```

Example Response

```
{
  "id" : 47,
  "jsonrpc" : "2.0",
  "result" :
  {
    "displayCapabilities" :
    {
       "displayType" : GEN2_8_DMA,
       "textFields" : [mainField1, mainField2, alertText1, alertText2,
alertText3, scrollableMessageBody, initialInteractionText, navigationText
1, navigationText2, audioPassThruDisplayText1,
audioPassThruDisplayText2, notificationText]
       "imageFields" :
       [
         {
          "name" : "softButtonImage",
          "imageTypeSupported" : ["GRAPHIC_BMP",
"GRAPHIC_JPEG", "GRAPHIC_PNG"],
          "imageResolution" :
           {
            "resolutionWidth" : 32,
            "resolutionHeight" : 32
           }
         },

         {
          "name" : "vrHelpItem",
          "imageTypeSupported" : ["GRAPHIC_BMP",
"GRAPHIC_JPEG", "GRAPHIC_PNG"],
          "imageResolution" :
           {
            "resolutionWidth" : 32,
            "resolutionHeight" : 32
           }
         },

         {
          "name" : "appIcon",
          "imageTypeSupported" : ["GRAPHIC_BMP",
"GRAPHIC_JPEG", "GRAPHIC_PNG"],
          "imageResolution" :
           {
            "resolutionWidth" : 64,
            "resolutionHeight" : 64
           }
         },
       ],

       "mediaClockFormats" : [CLOCK1, CLOCKTEXT4],
```

```
    "imageCapabilities" : [DYNAMIC],
    "graphicSupported" : true,
    "templatesAvailable" : ["DEFAULT", "MEDIA", "NON-MEDIA"],

    "screenParams" :
    {
     "resolution" :
      {
       "resolutionWidth" : 800,
       "resolutionHeight" : 480
      },

     "touchEventAvailable" :
      {
       "pressAvailable" : true,
       "multiTouchAvailable" : true,
       "doublePressAvailable" : false
      }
    },
    "numCustomPresetsAvailable" : 8
  },

  "buttonCapabilities" :
  [
   {
    "name" : OK,
    "shortPressAvailable" : true,
    "longPressAvailable" : true,
    "upDownAvailable" : true
   },
   {
    "name" : SEEKLEFT,
    "shortPressAvailable" : true,
    "longPressAvailable" : true,
    "upDownAvailable" : true
   },
   {
    "name" : SEEKRIGHT,
    "shortPressAvailable" : true,
    "longPressAvailable" : true,
    "upDownAvailable" : true
   },
   {
    "name" : TUNEUP,
    "shortPressAvailable" : true,
    "longPressAvailable" : true,
    "upDownAvailable" : true
   },
   {
    "name" : TUNEDOWN,
    "shortPressAvailable" : true,
    "longPressAvailable" : true,
    "upDownAvailable" : true
   },
```

```
    ],

    "softButtonCapabilities" :
    [
      {
        "shortPressAvailable" : true,
        "longPressAvailable" : true,
        "upDownAvailable" : true,
        "imageSupported" : true
      }
    ],

    "presetBankCapabilities" :
    {
      "onScreenPresetsAvailable" : true
    },

    "code" : 0,
    "method" : "UI.SetDisplayLayout"
  }
}
```

Example Error

```
{
  "id" : 47,
  "jsonrpc" : "2.0",
  "error" :
  {
    "code" : 6,
    "message" : "Ignored as the requested template is already
associated with the named appID",
    "data" :
    {
      "method" : "UI.SetDisplayLayout"
    }
  }
}
```

SetGlobalProperties

Type Function
Sender SDL
Purpose Set the UI properties of an application.

Description

SDL requests to set-up the data for VR help layout, the name and icon for in-application menu and the properties of the touchscreen keyboard.

The request may arrive for the application whether being active or in background on HMI (depends on Policy Table permissions applicable to mobile application request, by default allowed to operate in all HMI levels except of NONE).

The `vrHelp` parameter of the `SetGlobalProperties` RPC is used by the system to display the help items on the screen and the `helpPrompt` parameter is used by the system for playing out the associated TTS help prompt.

SDL sends SetGlobalProperties request with specific `<vrHelp>` and `<vrHelpTitle>` values to HMI in next cases:

1. If at any point in time, the application sends `SetGlobalProperties` RPC with the `vrHelp` **and** `helpPrompt` parameters, then SDL Core shall continue

with the existing behavior of forwarding such requests to HMI and SDL Core shall delete its internal list and stop sending `SetGlobalProperties` RPC to HMI after each AddCommmand/DeleteCommand request received from mobile.

2. If at any point in time, the application sends `SetGlobalProperties` RPC with **either** of `vrHelp` **or** `helpPrompt` parameters, then SDL Core shall continue with the existing behavior of forwarding such requests to HMI and SDL Core shall not delete its internal list and shall continue to update the parameter which was not provided by the application.
3. In case mobile app sends `AddCommand` with `CommandType = Command`, SDL must send update values of `vrHelp` via `SetGlobalProperties` to HMI. *(Note: AddCommand requests related to choice set must NOT trigger the update of "vrHelp")*
4. In case mobile app sends *SetGlobalProperties_request* to SDL:
    - with both valid values of *autoCompleteList* and *autoCompleteText* params, *SDL must*:
        - transfer *SetGlobalProperties_request* with *autoCompleteList* param and without (omitted) *autoCompleteText* param to HMI;
        - respond with `<resultCode_received_from_HMI>` to mobile app.
    - with valid *autoCompleteList* parameter with other valid params related to request and this request is allowed by Policies, *SDL must*:
        - transfer *SetGlobalProperties* with all requested params to HMI;
        - respond with `<resultCode_received_from_HMI>` to mobile app.
    - without *autoCompleteList* parameter with other valid params related to request and this request is allowed by Policies, *SDL must*:
        - transfer *SetGlobalProperties* with all requested parameters to HMI (*thus, without autoCompleteList*).
        - respond with `<resultCode_received_from _HMI>` to mobile app.

NOTE

By default `vrHelpTitle` value is set to application name.

Notes for HMI expected behavior:

1. The system shall have the ability to receive and store multiple strings for *autoCompleteText* per app.
2. When the system receives a new list of strings for autoCompleteText for a particular app, the system shall delete the previous list and replace it with the new list for that app.
3. When any of the keyboard layouts are being used, the system shall reference the list of *autoCompleteText* strings for that app.
4. As the user enters data on the keyboard, the system shall display the *autoCompleteText* strings which match the entry.
5. The number of matching *autoCompleteText* strings displayed shall only be limited by the character length constraints of the hmi.
6. The system shall provide the user the ability to select one of the displayed matching *autoCompleteText* strings without having to enter the entire string.
7. When the user selects one of the displayed matching *autoCompleteText* string, the system shall submit that entry and not require further user input for submission.

REQUEST

BEHAVIOR

MUST

1. Store the information and associate it with appID.
    - Note: *Initially, the appID together with other application-related information is provided by SDL within UpdateAppList or OnAppRegistered RPCs.*
2. Whenever the User activates VR, set up the requested values for VR help layout, the name and icon for in-application menu and the properties of the touchscreen keyboard (if supported):
    - display the list of commands available for voice recognition. SDL provides the title for this list (*vrHelpTitle* parameter) and the list of commands itself (vrHelp parameter which is an array of VrHelpItem‘s).
    - display the in-application menu for every active application on User’s request. It must contain SDL-requested commands (UI.AddCommand) and sub menus (UI.AddSubMenu). SDL provides the values for the name (*menuTitle* parameter) and for the icon (menuIcon parameter) of this in-application menu. The values for in-application menu and touchscreen keyboard are allowed by SDL for navigation type of application only.
    - display the onscreen keyboard upon User`s request within the following condition: all keyboardProperties supported by HMI must be embodied in HMI_capabilities.json file. In this case SDL are able to compare keyboardProperties requested by mobile device with actual supported keyboardProperties and send to HMI only that are supported.
    - use default keyboardProperties – parameter in case SDL transfers UI.SetGlobalProperties request with omitted or empty keyboardProperties param to HMI.
    - ***Important Note:*** *If HMI-defined VR commands are accessible together with those provided by SDL via VR.AddCommand, HMI must:*
        - *add the corresponding VR HMI-defined commands to the list of VR help items provided by SDL via UI.SetGlobalProperties*
        - *display the complete list of available VR commands (SDL-defined and HMI-defined ones) when the User activates VR.*
3. Respond to the request.

PARAMETERS

NAME	TYPE	MANDATORY	ADDITIONAL
vrHelpTitle	String	false	maxlength: 500
vrHelp	Common.VrHelpItem	false	array: true minsize: 1 maxsize: 100
menuTitle	String	false	maxlength: 500
menuIcon	Common.Image	false	-
keyboardProperties	Common.KeyboardProperties	false	-
appID	Integer	true	-

RESPONSE

RESULT	DESCRIPTION	MESSAGE TYPE WEBSOCKET	MESSAGE TYPE D-BUS	MESSAGE PARAMS
Success	SUCCESS: HMI has set the requested properties.	JSON response	Method return	code: 0
Failure	INVALID_ID:appID is not valid (e.g.does not exist)	JSON response	Method return	code: 13
Failure	INVALID_DATA: The data sent is invalid (invalid JSON syntax or parameters out of bounds or of wrong type)	JSON response	Method return	code: 11
Failure	GENERIC_ERROR: The unknown issue occurred or other codes are not applicable.	JSON response	Method return	code: 22

NOTE

In case HMI does not respond SDL's request during SDL-default timeout (10 sec), SDL will return GENERIC_ERROR result code to the corresponding mobile app's request. Please see Result Enumeration for all SDL-supported codes.

PARAMETERS

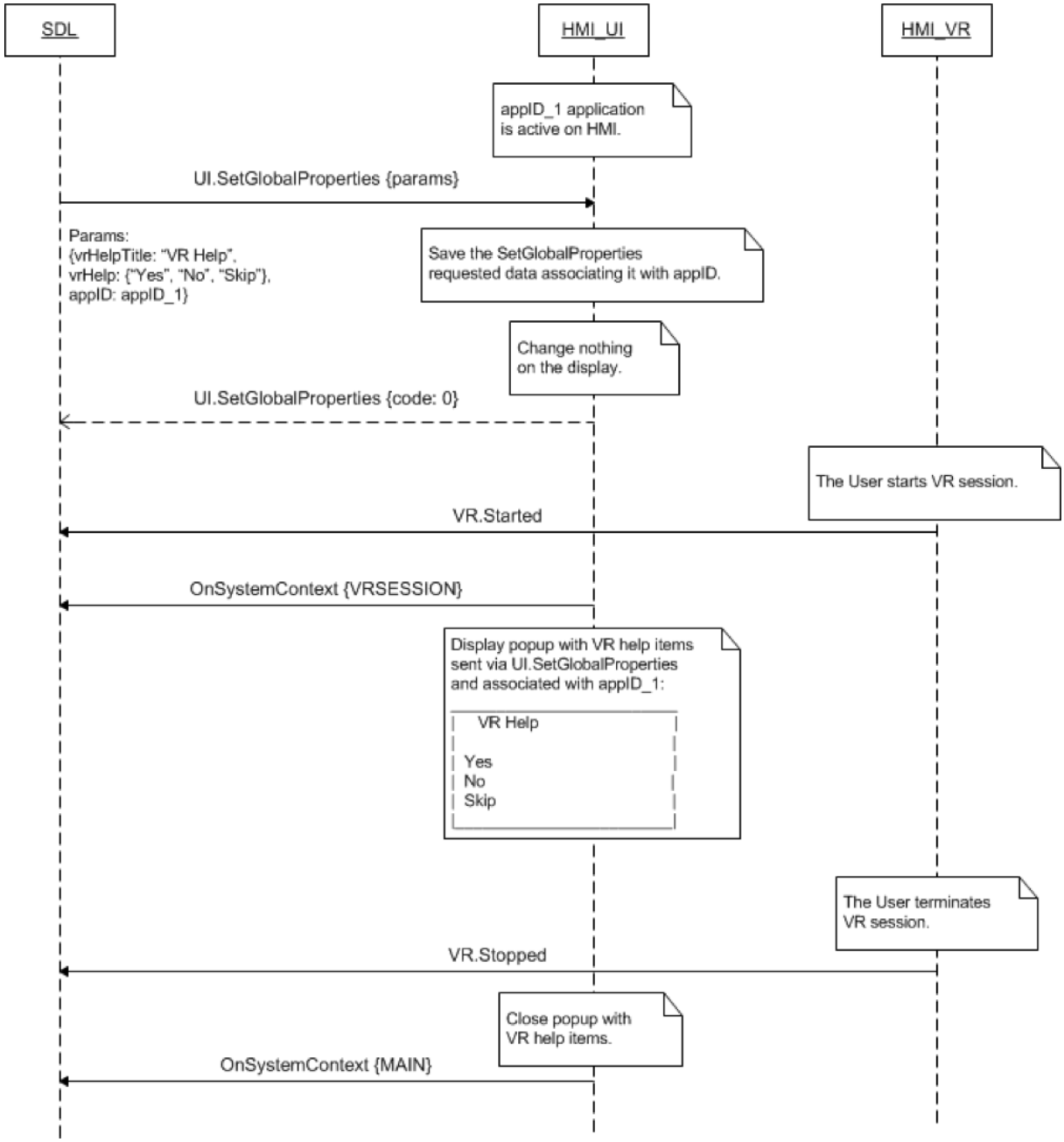
This RPC has no additional parameter requirements

Sequence Diagrams

SEQUENCE DIAGRAM

SetGlobalProperties for active app on HMI with VR activation

View Diagram


SDL
HMI_UI
HMI_VR
appID_1 application is active on HMI.
UI.SetGlobalProperties {params}
Params:
{vrHelpTitle: "VR Help",
vrHelp: {"Yes", "No", "Skip"},
appID: appID_1}
Save the SetGlobalProperties requested data associating it with appID.
Change nothing on the display.
UI.SetGlobalProperties {code: 0}
The User starts VR session.
VR.Started
OnSystemContext {VRSESSION}
Display popup with VR help items sent via UI.SetGlobalProperties and associated with appID_1:
VR Help
Yes
No
Skip
The User terminates VR session.
VR.Stopped
Close popup with VR help items.
OnSystemContext {MAIN}

Example Request

```
{
 "id" : 116,
 "jsonrpc" : "2.0",
 "method" : "UI.SetGlobalProperties",
 "params" :
 {
  "vrHelpTitle" : "Choose the action",
  "vrHelp" :
  [
      {
       "text" : "Pause",
       "image" :
       [
          "value" : "tmp/SDL/app/Pandora/icon_1067.jpg",
          "imageType" : DYNAMIC
       ],
       "position" : 1
      },
      {
       "text" : "Resume",
       "image" :
       [
          "value" : "tmp/SDL/app/Pandora/icon_1083.jpeg",
          "imageType" : DYNAMIC
       ],
       "position" : 2
      },
      {
       "text" : "Skip",
       "image" :
       [
          "value" : "tmp/SDL/app/Pandora/icon_1013.jpeg",
          "imageType" : DYNAMIC
       ],
       "position" : 3
      },
      {
       "text" : "Bookmark",
       "image" :
       [
          "value" : "tmp/SDL/app/Pandora/icon_1046.jpeg",
          "imageType" : DYNAMIC
       ],
       "position" : 4
      }
  ],
  "appID" : 53880
 }
}
```

Example Response

```
{
 "id" : 116,
 "jsonrpc" : "2.0",
 "result" :
 {
  "code" : 0,
  "method" : "UI.SetGlobalProperties"
 }
}
```

Example Error

```
{
 "id" : 116,
 "jsonrpc" : "2.0",
 "error" :
 {
  "code" : 11,
  "message" : "Invalid data",
  "data" :
  {
   "method" : "UI.SetGlobalProperties"
  }
 }
}
```

SetMediaClockTimer

Type Function
Sender SDL
Purpose Set a value and update mode for the media clock of a media application.

The UI.SetMediaClock timer request indicates either an initial value for the media clock timer for a media application or an update to this value. The request may come for the application which is not currently active on the HMI.

REQUEST

MUST

1. Perform the update type indicated by the `updateMode` parameter:
    - If the application is not active, the HMI must still store the values to be calculated for later display on the HMI.
    - If the application is active, the updates must begin immediately.

2. Exhibit the following behavior based on the `updateMode` parameter:
    - COUNTUP/COUNTDOWN modes:
        - Start counting up or down from the requested `startTime` value with a step of 1 second;
        - Continue counting up or down until:
            - The next request of *SetMediaClockTimer* with appropriate parameters comes;
            - Zero is reached in the case of COUNTDOWN.
    - PAUSE mode:
        - Pause the timer that is counting up or down;
        - If `startTime` or `endTime` parameters are provided, the values must be updated on the HMI.
    - CLEAR mode:
        - Clear `startTime` to 00:00:00 in the case that the `startTime` parameter is not provided in the request, otherwise, `startTime` must be updated with a new value. It is up to HMI to determine the way the media clock timer is cleared: either to remove it from display or to set it to zero.
3. Respond with the result code (see *response* section) correspondingly to the results of this RPC execution.

NOTE

1. SDL will not send this request if the `mediaClock` field is not indicated as supported in UI.GetCapabilities.
2. HMI must remember the mode and the value (continuing to update it in case of COUNTUP / COUNTDOWN) of the media clock timer associated with appID and display the accurate values whenever the appID application is activated after having been deactivated.
3. Initially, the appID together with other application-related information is provided by SDL within one of *UpdateAppList* and *OnAppRegistered* RPCs.

PARAMETERS

NAME	TYPE	MANDATORY	DESCRIPTION
startTime	Common.TimeFormat	false	*startTime* must be provided together with modes: "COUNTUP", "COUNTDOWN", "PAUSE" to HMI. *startTime* will be ignored for "RESUME", and "CLEAR".
endTime	Common.TimeFormat	false	endTime can be provided together with modes: "COUNTUP", "COUNTDOWN", "PAUSE" to HMI. To be used to calculate any visual progress bar (if not provided, this feature is ignored). If endTime is greater then startTime for COUNTDOWN or less than startTime for COUNTUP, then the request will return an INVALID_DATA. *endTime* will be ignored for "PAUSE", "RESUME", and "CLEAR".

NAME	TYPE	MANDATORY	DESCRIPTION
updateMode	Common.ClockUpdateMode	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	Common.AudioStreamingIndicator	false	Indicates that a button press of the Play/Pause button would play, pause or stop the current playback.
appID	Integer	true	

RESPONSE

PARAMETERS

Success	SUCCESS: 1. HMI has started counting up or down the media clock from the requested value (in case of COUNTUP or COUNTDOWN requested mode). 2. HMI has paused the media clock value (PAUSE requested mode). 3. HMI has resumed the media clock in the mode that had been in effect before pausing, i.e. has started counting up or	JSON response	Method return	code: 0	HMI must respond right after the action requested with update Mode parameter has been performed.

RESULT	DESCRIPTION	MESSAGE TYPE WEBSOCKET	MESSAGE TYPE D-BUS	MESSAGE PARAMS	NOTES
Failure	IGNORED: 1. Request with RESUME mode arrives when the media clock is counting, already resumed or cleared with the previous request. 2. Request with PAUSE mode arrives when the media clock is paused or cleared with the previous request.	JSON error message	Method return	code: 6	Applicable to this RPC result codes.

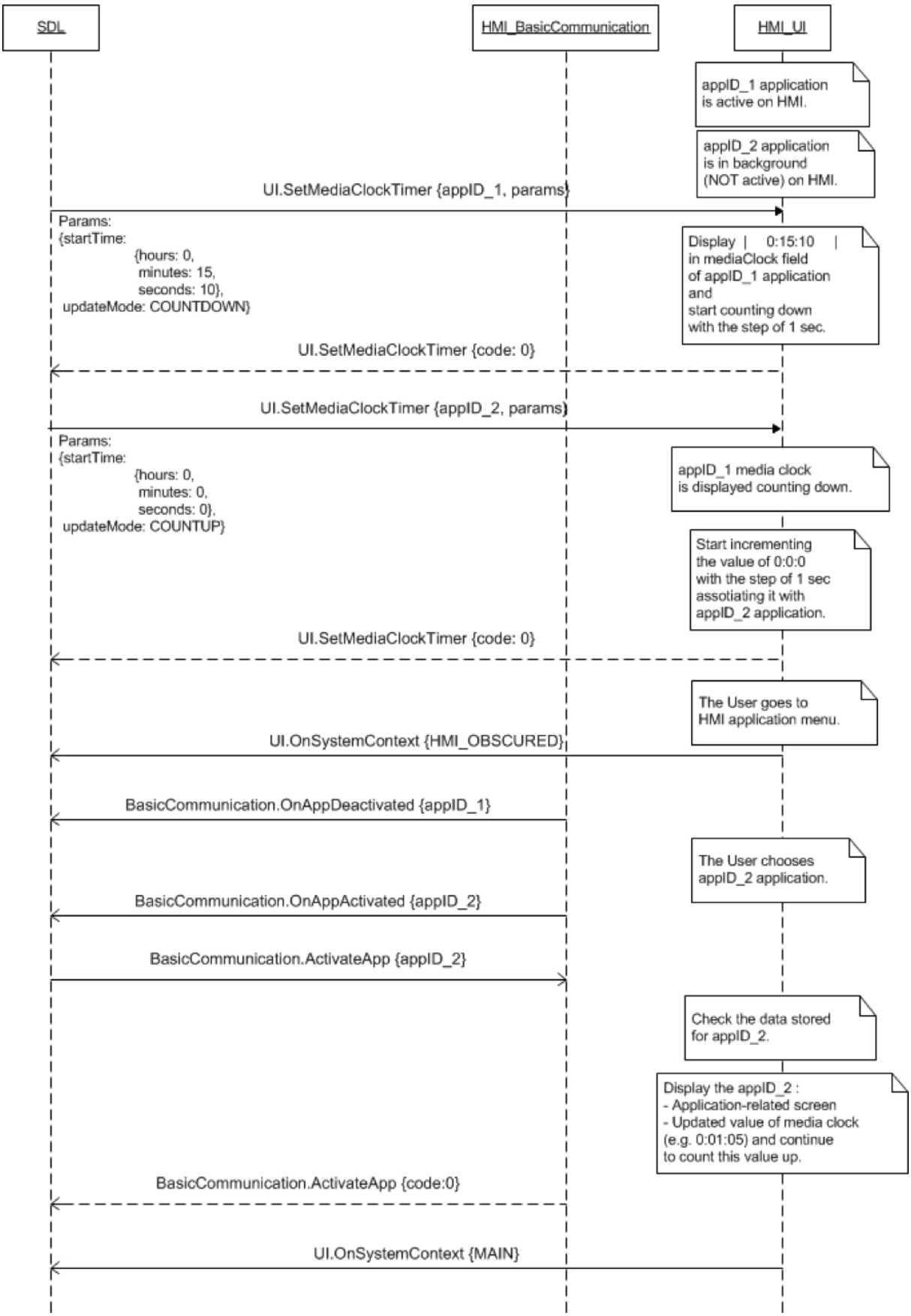
RESULT	DESCRIPTION	MESSAGE TYPE WEBSOCKET	MESSAGE TYPE D-BUS	MESSAGE PARAMS	NOTES
Failure	INVALID_DATA: The data sent is invalid (invalid JSON syntax or parameters out of bounds or of wrong type)	JSON error message	Method return	code: 11	Applicable to this RPC result codes.
Failure	INVALID_ID: appID is invalid (e.g. doesn't exist)	JSON error message	Method return	code: 13	Applicable to this RPC result codes.
Failure	GENERIC_ERROR: The unknown issue occurred or other codes are not applicable.	JSON error message	Method return	code: 22	Applicable to this RPC result codes.

Sequence Diagrams

SEQUENCE DIAGRAM

SetMediaClockTimer COUNTUP and COUNTDOWN for Full and Background applications

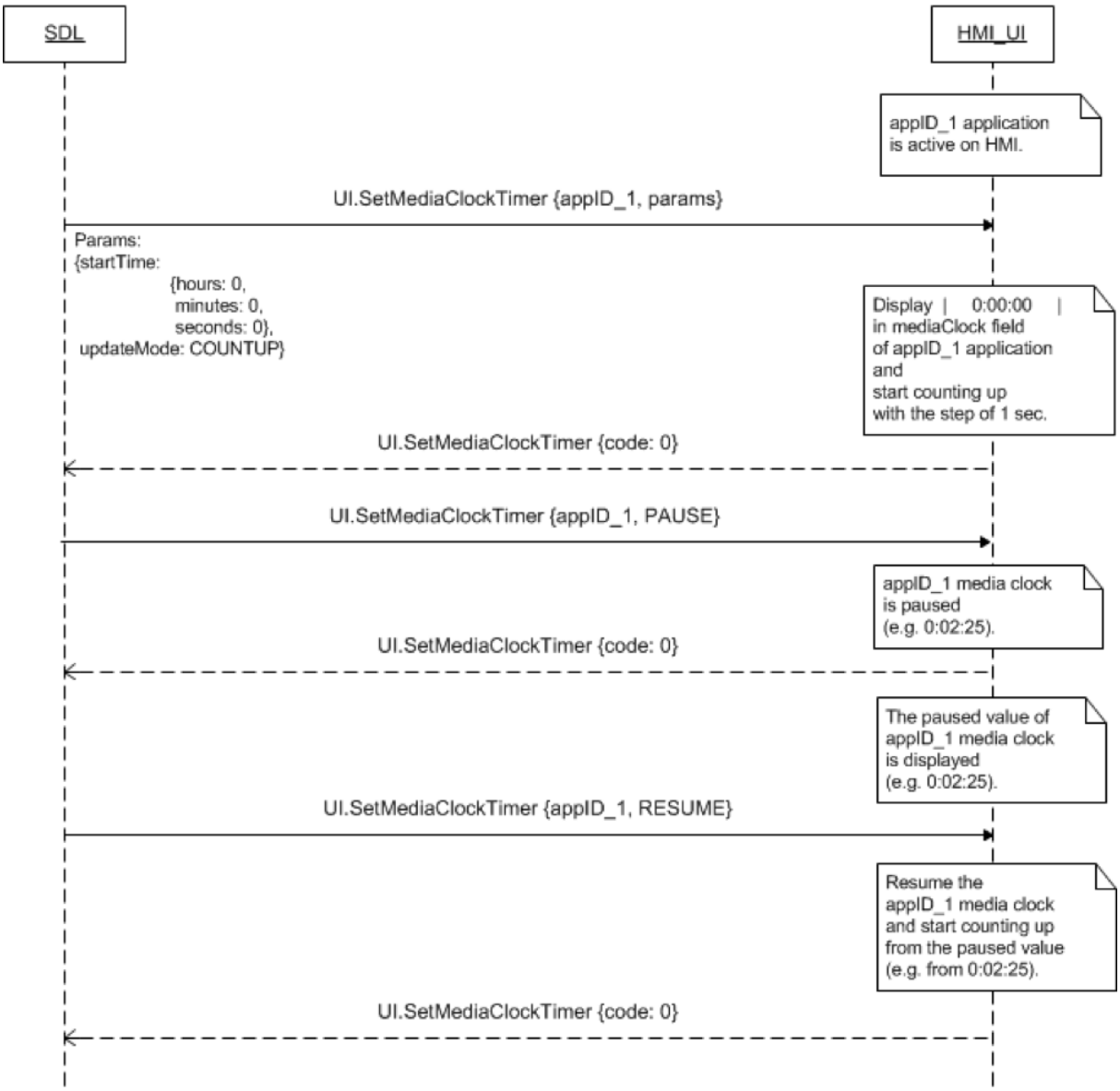
View Diagram



SEQUENCE DIAGRAM

SetMediaClockTimer Pause and Resume for Active Application

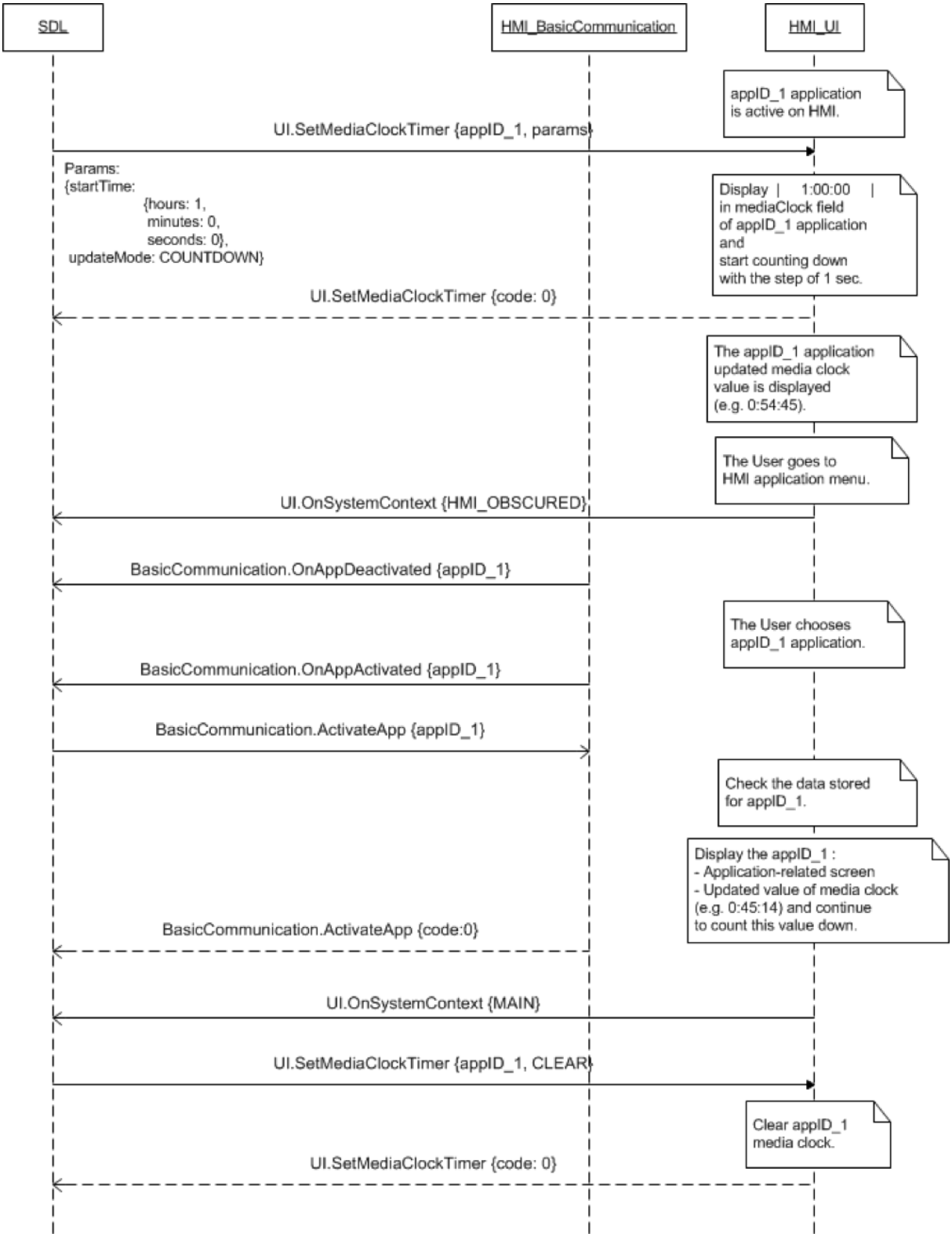
View Diagram



SEQUENCE DIAGRAM

SetMediaClockTimer COUNTDOWN for a deactivated application

View Diagram


SDL
HMI_BasicCommunication
HMI_UI
appID_1 application is active on HMI.
UI.SetMediaClockTimer {appID_1, params}
Params:
{startTime:
{hours: 1,
minutes: 0,
seconds: 0},
updateMode: COUNTDOWN}
Display | 1:00:00 | in mediaClock field of appID_1 application and start counting down with the step of 1 sec.
UI.SetMediaClockTimer {code: 0}
The appID_1 application updated media clock value is displayed (e.g. 0:54:45).
The User goes to HMI application menu.
UI.OnSystemContext {HMI_OBSCURED}
BasicCommunication.OnAppDeactivated {appID_1}
The User chooses appID_1 application.
BasicCommunication.OnAppActivated {appID_1}
BasicCommunication.ActivateApp {appID_1}
Check the data stored for appID_1.
Display the appID_1 :
- Application-related screen
- Updated value of media clock (e.g. 0:45:14) and continue to count this value down.
BasicCommunication.ActivateApp {code:0}
UI.OnSystemContext {MAIN}
UI.SetMediaClockTimer {appID_1, CLEAR}
Clear appID_1 media clock.
UI.SetMediaClockTimer {code: 0}

Example Request

```
{
 "id" : 109,
 "jsonrpc" : "2.0",
 "method" : "UI.SetMediaClockTimer",
 "params" :
 {
  "startTime" :
  {
     "hours" : 0,
     "minutes" : 18,
     "seconds" : 17
  },
  "updateMode" : "COUNTUP",
  "audioStreamingIndicator" : "PAUSE",
  "appID" : 65146
 }
}
```

Example Response

```
{
 "id" : 109,
 "jsonrpc" : "2.0",
 "result" :
 {
  "code" : 0,
  "method" : "UI.SetMediaClockTimer"
 }
}
```

Example Error


```
{
 "id" : 109,
 "jsonrpc" : "2.0",
 "error" :
 {
  "code" : 11,
  "message" : "Invalid data",
  "data" :
  {
   "method" : "UI.SetMediaClockTimer"
  }
 }
}
```


Show

Type Function
Sender SDL
Purpose Update fields displayed on the HMI for the specified application.

REQUEST

PARAMETERS

NAME	TYPE	MANDATORY	ADDITIONAL
showStrings	Common.TextFieldStruct	true	array: true minsize: 0 maxsize: 7
alignment	Common.TextAlignment	false	
graphic	Common.Image	false	
secondaryGraphic	Common.Image	false	
softButtons	Common.SoftButton	false	array: true minsize: 0 maxsize: 8
customPresets	String	false	array: true minsize: 0 maxsize: 10 maxlength: 500
appID	Integer	true	

RESPONSE

PARAMETERS

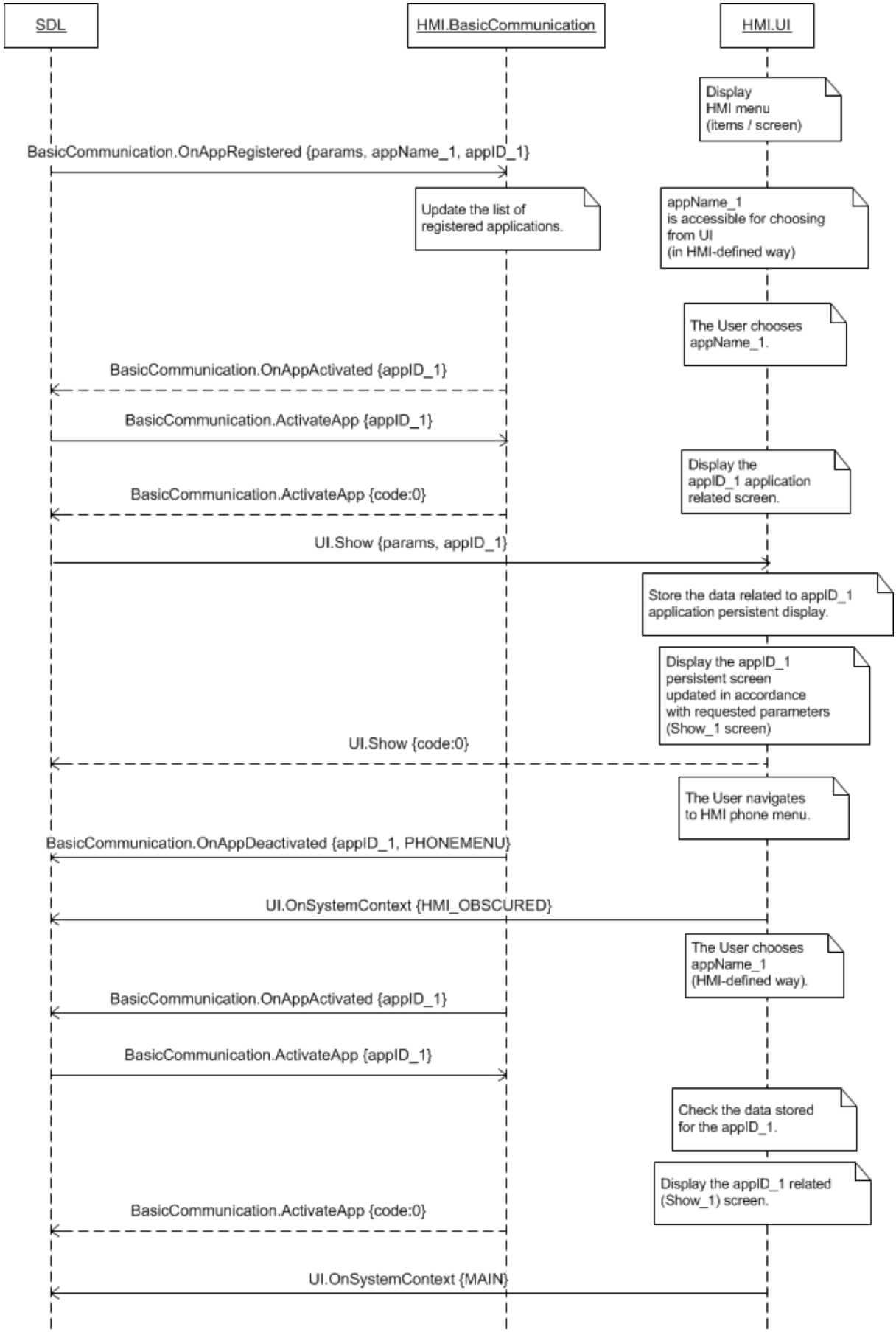
This RPC has no additional parameter requirements

Sequence Diagrams

SEQUENCE DIAGRAM

Active App shows and is deactivated then reactivated

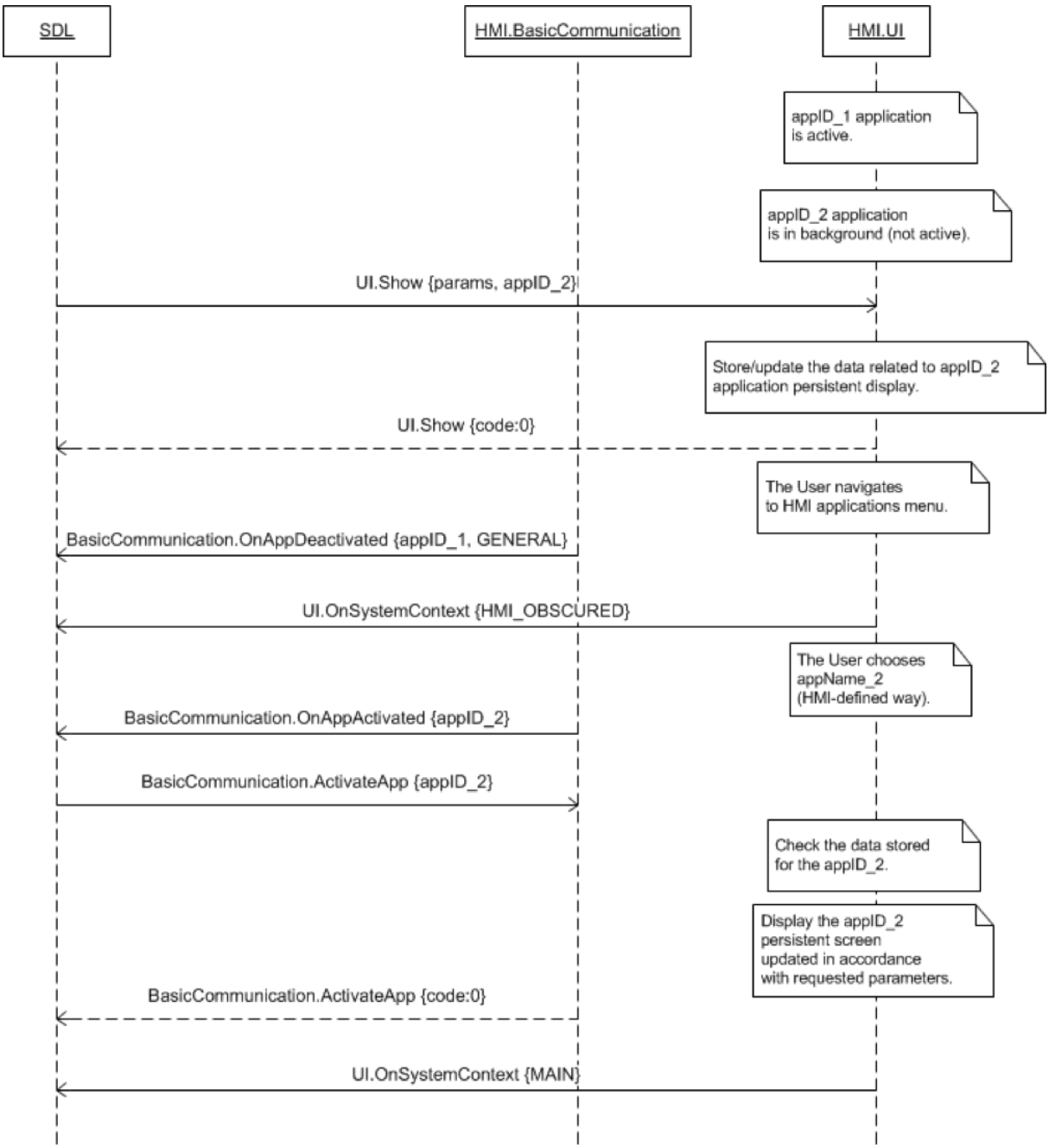
View Diagram


SDL
HMI.BasicCommunication
HMI.UI
Display HMI menu (items / screen)
BasicCommunication.OnAppRegistered {params, appName_1, appID_1}
Update the list of registered applications.
appName_1 is accessible for choosing from UI (in HMI-defined way)
The User chooses appName_1.
BasicCommunication.OnAppActivated {appID_1}
BasicCommunication.ActivateApp {appID_1}
Display the appID_1 application related screen.
BasicCommunication.ActivateApp {code:0}
UI.Show {params, appID_1}
Store the data related to appID_1 application persistent display.
Display the appID_1 persistent screen updated in accordance with requested parameters (Show_1 screen)
UI.Show {code:0}
The User navigates to HMI phone menu.
BasicCommunication.OnAppDeactivated {appID_1, PHONEMENU}
UI.OnSystemContext {HMI_OBSCURED}
The User chooses appName_1 (HMI-defined way).
BasicCommunication.OnAppActivated {appID_1}
BasicCommunication.ActivateApp {appID_1}
Check the data stored for the appID_1.
Display the appID_1 related (Show_1) screen.
BasicCommunication.ActivateApp {code:0}
UI.OnSystemContext {MAIN}

SEQUENCE DIAGRAM

Inactive App Sends Show

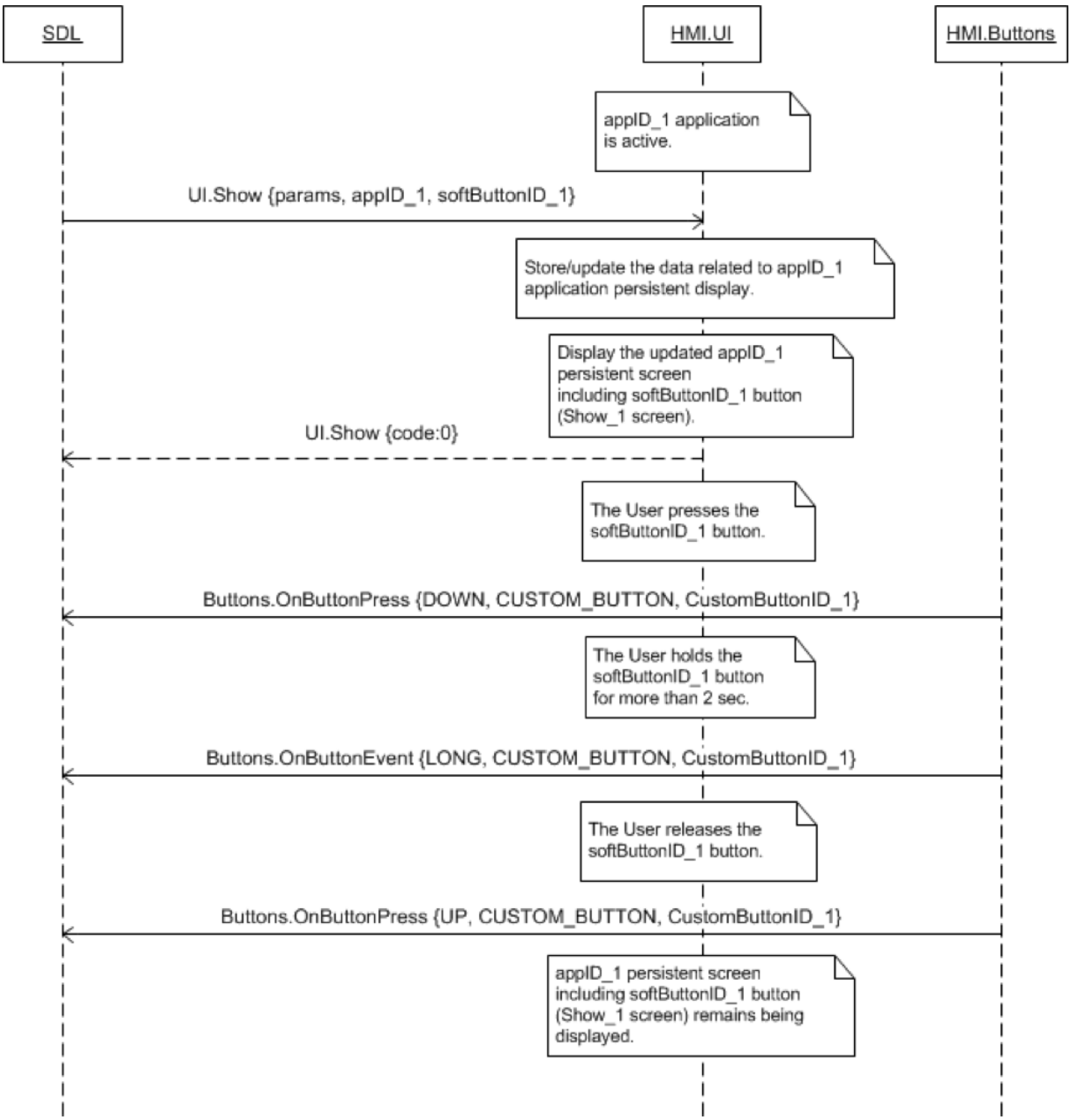
View Diagram



SEQUENCE DIAGRAM

Show with Soft Buttons

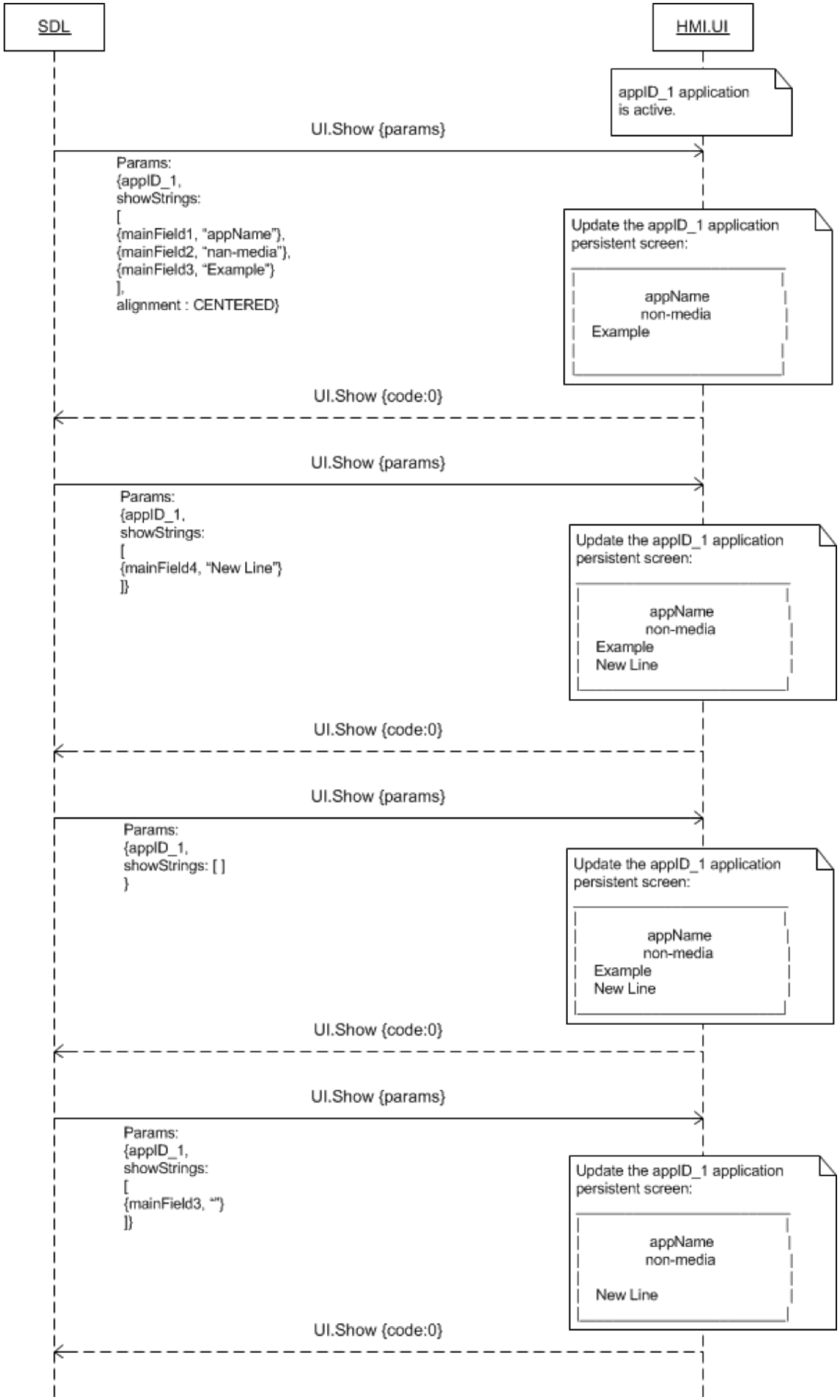
View Diagram



SEQUENCE DIAGRAM

Show Text Fields

View Diagram


SDL
HMI.UI
appID_1 application is active.
UI.Show {params}
Params:
{appID_1,
showStrings:
[
{mainField1, "appName"},
{mainField2, "nan-media"},
{mainField3, "Example"}
],
alignment : CENTERED}
Update the appID_1 application persistent screen:
appName
non-media
Example
UI.Show {code:0}
UI.Show {params}
Params:
{appID_1,
showStrings:
[
{mainField4, "New Line"}
]}
Update the appID_1 application persistent screen:
appName
non-media
Example
New Line
UI.Show {code:0}
UI.Show {params}
Params:
{appID_1,
showStrings: [ ]
}
Update the appID_1 application persistent screen:
appName
non-media
Example
New Line
UI.Show {code:0}
UI.Show {params}
Params:
{appID_1,
showStrings:
[
{mainField3, ""}
]}
Update the appID_1 application persistent screen:
appName
non-media
New Line
UI.Show {code:0}

Example Request

```
{
 "id" : 120,
 "jsonrpc" : "2.0",
 "method" : "UI. Show",
 "params" :
 {
  "showStrings" :
  [
   {
    "fieldName" : mainField1,
    "fieldText" : "Favourite Album"
   },
   {
    "fieldName" : mediaClock,
    "fieldText" : "1:45:12"
   },
   {
    "fieldName" : mediaTrack,
    "fieldText" : "Ironic – The Collection – Alanis Morissette"
   }
  ],
  "alignment" : LEFT_ALIGNED,
  "graphic" :
  {
     "value" : "tmp/SDL/app/Best_Media/AM-Collection-cover.png",
     "imageType" : DYNAMIC
  },
  "softButtons" :
  [
   {
   "type" :  BOTH,
   "text" : "Change Album",
   "image" :
    [
     "value" : "tmp/SDL/app/Best_Media/change_alb_icon.jpg",
     "imageType" : DYNAMIC
    ],
   "softButtonID" : 48,
   "systemAction" :  DEFAULT_ACTION
   },

   {
   "type" :  TEXT,
   "text" : "Change Artist",
   "softButtonID" : 57
   },
  ],
  "customPresets" : ["Like Song", "Like Album"],
  "appID" : 8726
```

```
  }
}
```

Example Response

```
{
 "id" : 120,
 "jsonrpc" : "2.0",
 "result" :
 {
  "code" : 0,
  "method" : "UI.Show"
 }
}
```

Example Error

```
{
 "id" : 120,
 "jsonrpc" : "2.0",
 "error" :
 {
  "code" : 22,
  "message" : "An unknown issue occurred ",
  "data" :
  {
   "method" : "UI.Show"
  }
 }
}
```

ShowCustomForm

REQUEST

PARAMETERS

NAME	TYPE	MANDATORY	ADDITIONAL
customFormID	String	true	maxlength: 500
parentFormID	String	false	maxlength: 500

RESPONSE

PARAMETERS

NAME	TYPE	MANDATORY	ADDITIONAL
info	String	false	maxlength: 1000

Example Request

Example Response

Example Error

Slider

Type Function
Sender SDL
Purpose Display a user controlled slider UI element.

REQUEST

PARAMETERS

NAME	TYPE	MANDATORY	ADDITIONAL
numTicks	Integer	true	minvalue: 2 maxvalue: 26
position	Integer	true	minvalue: 1 maxvalue: 26
sliderHeader	String	true	maxlength: 500
sliderFooter	String	false	array: true minsize: 1 maxsize: 26 maxlength: 500
timeout	Integer	true	minvalue: 1000 maxvalue: 65535
appID	Integer	true	

RESPONSE

PARAMETERS

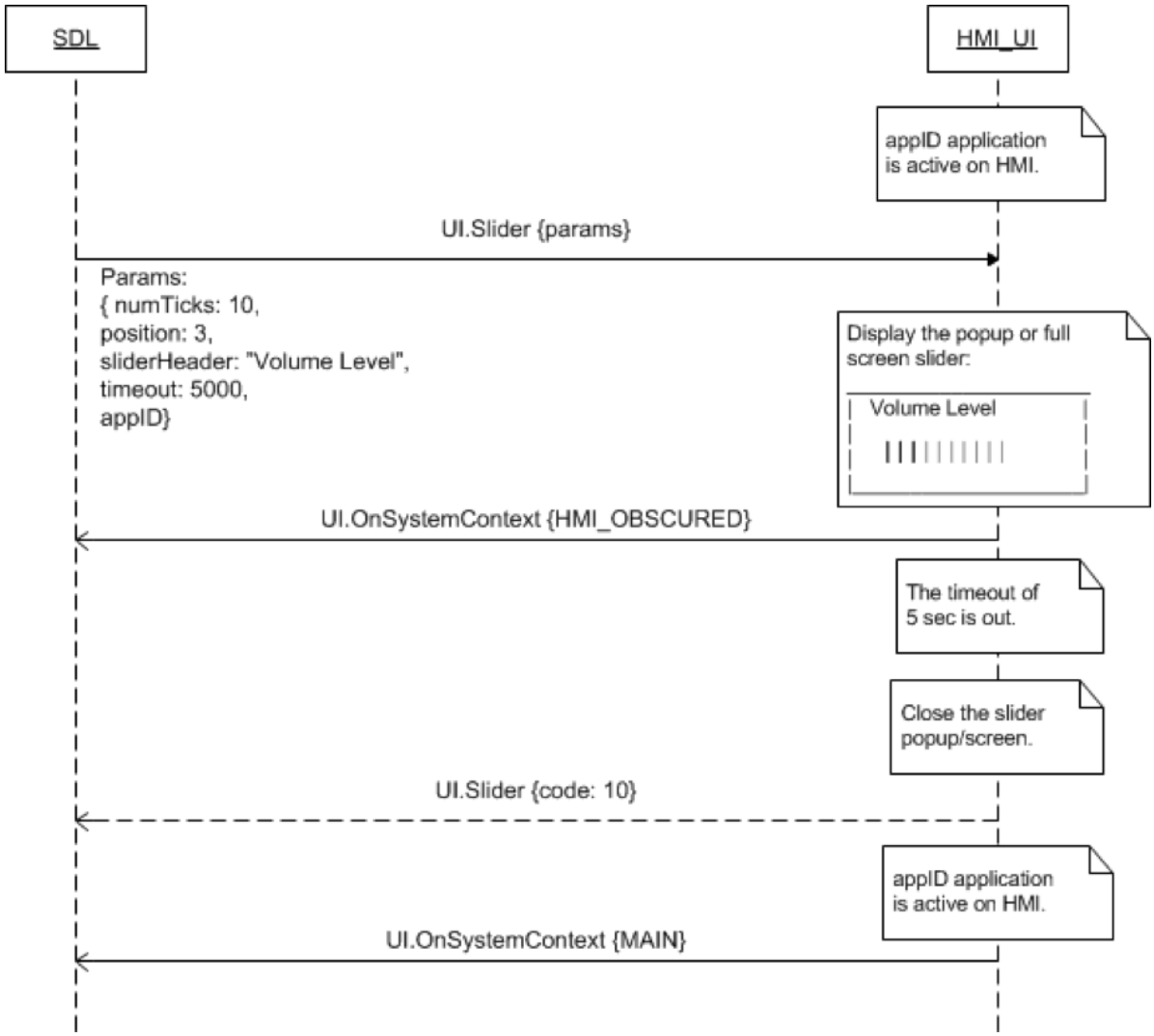
NAME	TYPE	MANDATORY	ADDITIONAL
sliderPosition	Integer	false	minvalue: 1 maxvalue: 26

Sequence Diagrams

SEQUENCE DIAGRAM

Slider with static footer displayed closed by timeout

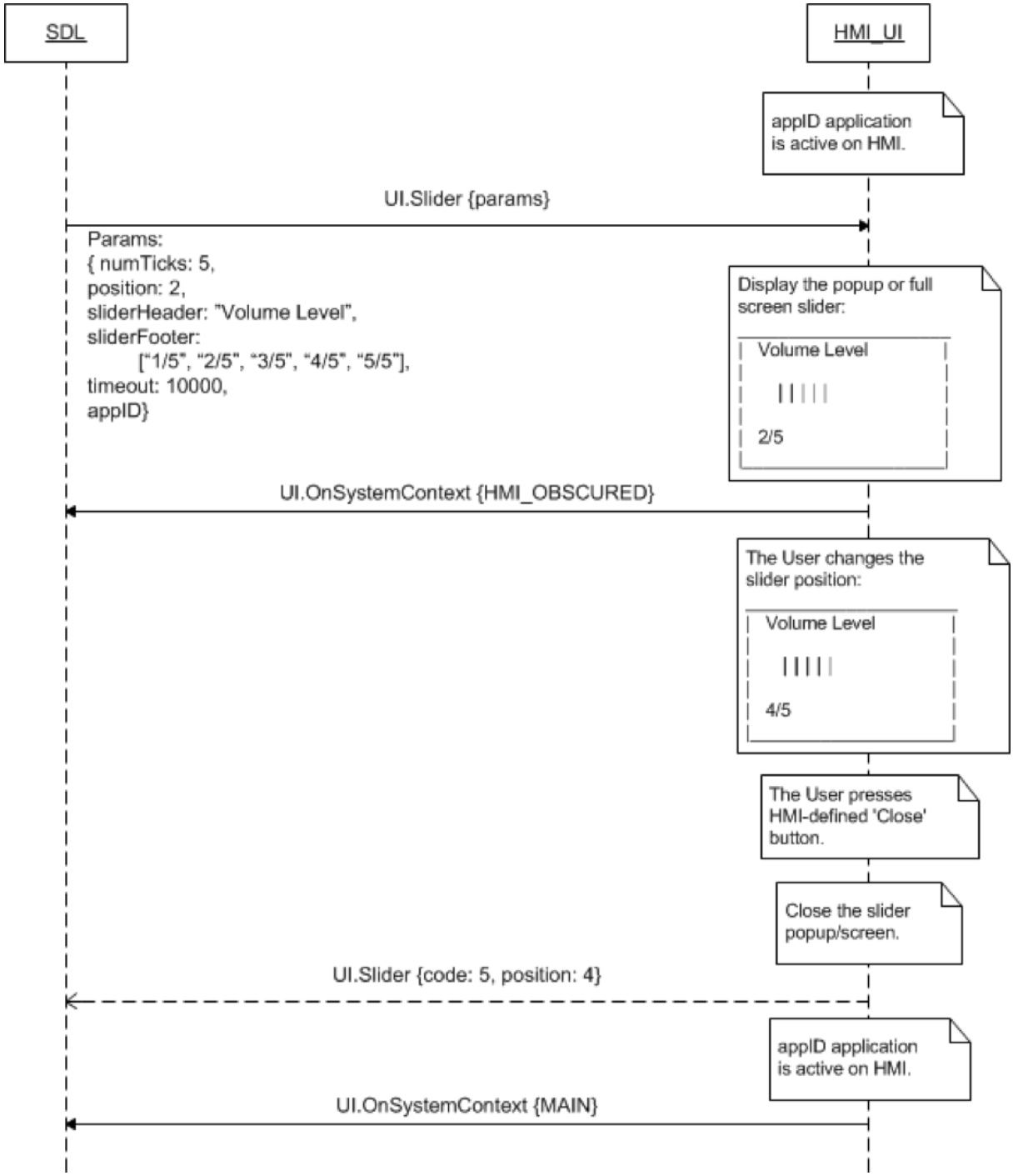
View Diagram



SEQUENCE DIAGRAM

Slider with dynamic footer aborted

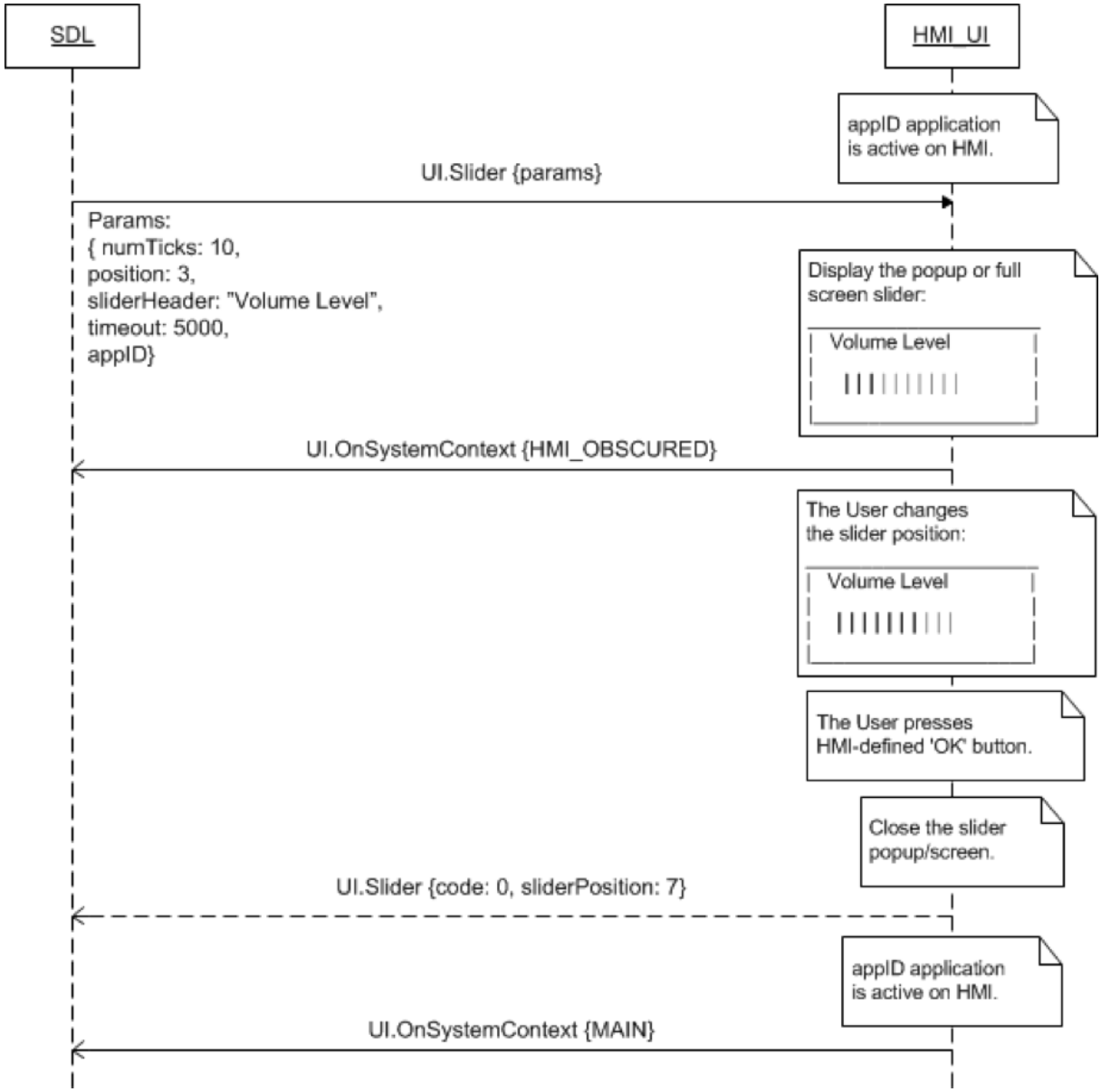
View Diagram



SEQUENCE DIAGRAM

Slider with OK Button press

View Diagram


SDL
HMI_UI
appID application is active on HMI.
UI.Slider {params}
Params:
{ numTicks: 10,
position: 3,
sliderHeader: "Volume Level",
timeout: 5000,
appID}
Display the popup or full screen slider:
Volume Level
UI.OnSystemContext {HMI_OBSCURED}
The User changes the slider position:
Volume Level
The User presses HMI-defined 'OK' button.
Close the slider popup/screen.
UI.Slider {code: 0, sliderPosition: 7}
appID application is active on HMI.
UI.OnSystemContext {MAIN}

Example Request

```
{
 "id" : 133,
 "jsonrpc" : "2.0",
 "method" : "UI.Slider",
 "params" :
 {
  "numTicks" : 5,
  "position" : 2,
  "sliderHeader" : "Volume Level",
  "sliderFooter" : [ "1/5", "2/5", "3/5", "4/5", "5/5" ],
  "timeout" : 10000,
  "appID" : 4328
 }
}
```

Example Response

```
{
 "id" : 133,
 "jsonrpc" : "2.0",
 "result" :
 {
  "sliderPosition" : 4,
  "code" : 0,
  "method" : "UI.Slider"
 }
}
```

Example Error

```
{
 "id" : 133,
 "jsonrpc" : "2.0",
 "error" :
 {
  "code" : 5,
  "message" : "A command was aborted due to user interaction",
  "data" :
  {
   "sliderPosition" : 5
   "method" : "UI.Slider"
  }
 }
}
```

```
{
 "id" : 133,
 "jsonrpc" : "2.0",
 "error" :
 {
  "code" : 13,
  "message" : "One of the provided IDs is not valid",
  "data" :
  {
   "method" : "UI.Slider"
  }
 }
}
```

DiagnosticMessage

Type Function
Sender SDL
Purpose

Request current diagnostic messages.

REQUEST

PARAMETERS

NAME	TYPE	MANDATORY	ADDITIONAL
targetID	Integer	true	minvalue: 0 maxvalue: 65535
messageLength	Integer	true	minvalue: 0 maxvalue: 65535
messageData	Integer	true	array: true minsize: 1 maxsize: 65535 minvalue: 0 maxvalue: 255
appID	Integer	true	

RESPONSE

PARAMETERS

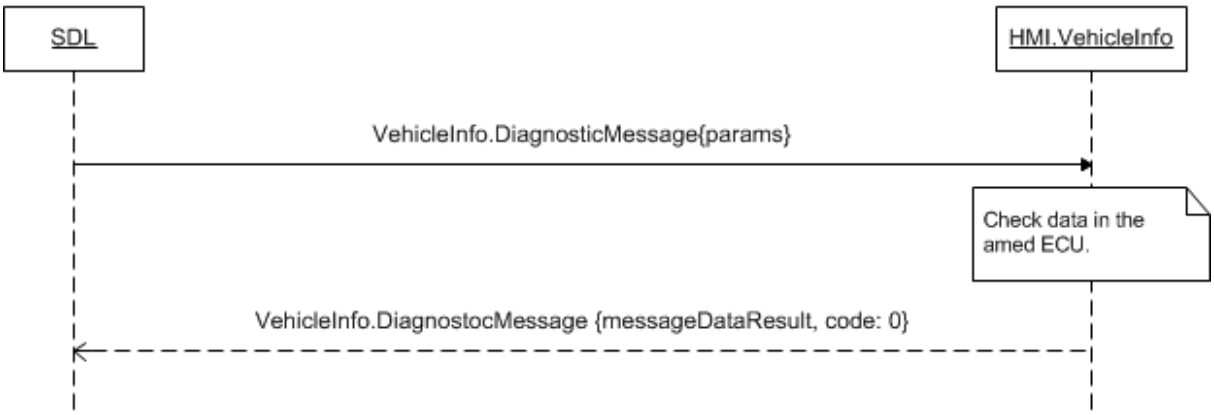
NAME	TYPE	MANDATORY	ADDITIONAL
messageDataResult	Integer	true	array: true minsize: 1 maxsize: 65535 minvalue: 0 maxvalue: 255

Sequence Diagrams

SEQUENCE DIAGRAM

DiagnosticMessage

View Diagram



Example Request

```
{
 "id" : 139,
 "jsonrpc" : "2.0",
 "method" : "VehicleInfo.DiagnosticMessage",
 "params" :
 {
  "targetID" : 5456
  "messageLength" : 1084,
  "messageData" : [1,2,3,4,5,6,7,8,9]
 }
}
```

Example Response

```
{
 "id" : 139,
 "jsonrpc" : "2.0",
 "result" :
 {
  "messageDataResult" : [1,2,3,4,5,6],
  "code" : 0,
  "method" : "VehicleInfo.GetDTCs"
 }
}
```

Example Error


```
{
 "id" : 139,
 "jsonrpc" : "2.0",
 "error" :
 {
  "code" : 9,
  "message" : "Data not available",
  "data" :
  {
   "method" : "VehicleInfo.GetDTCs"
  }
 }
}
```


GetDTCs

Type Function
Sender SDL
Purpose Request DTCs from vehicle.

REQUEST

PARAMETERS

NAME	TYPE	MANDATORY	ADDITIONAL
ecuName	Integer	true	minvalue: 0 maxvalue: 65535
dtcMask	Integer	false	minvalue: 0 maxvalue: 255
appID	Integer	true	

RESPONSE

PARAMETERS

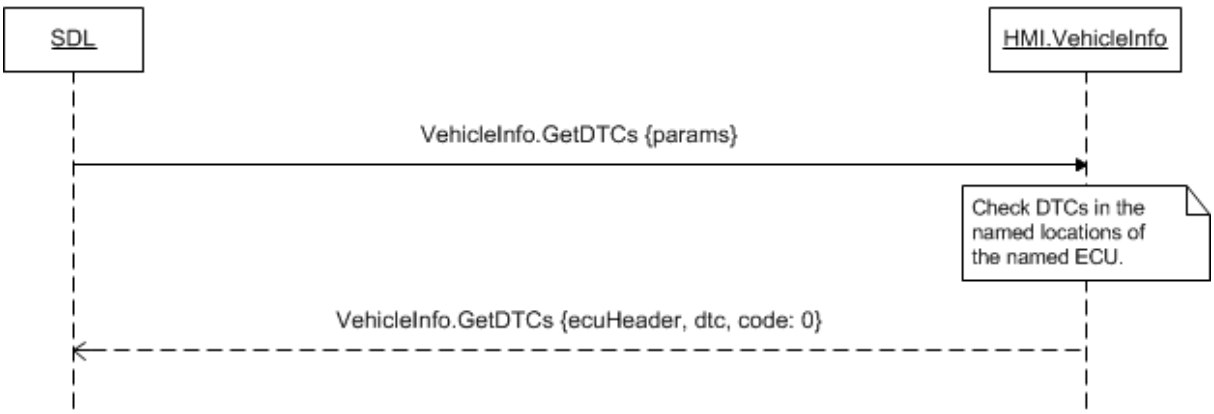
NAME	TYPE	MANDATORY	ADDITIONAL
ecuHeader	Integer	true	minvalue: 0 maxvalue: 65535
dtc	String	false	array: true minsize: 1 maxsize: 15 maxlength: 10

Sequence Diagrams

SEQUENCE DIAGRAM

GetDTCs

View Diagram



Example Request

```
{
 "id" : 139,
 "jsonrpc" : "2.0",
 "method" : "VehicleInfo.GetDTCs",
 "params" :
 {
  "ecuName" : 56
  "dtcMask" : 84,
  "appID" : 65645
 }
}
```

Example Response


```
{
 "id" : 139,
 "jsonrpc" : "2.0",
 "result" :
 {
  "ecuHeader" : 6534,
  "dtc" : ["84752093", "28237", "748398"],
  "code" : 0,
  "method" : "VehicleInfo.GetDTCs"
 }
}
```


Example Error


```
{
 "id" : 139,
 "jsonrpc" : "2.0",
 "error" :
 {
  "code" : 9,
  "message" : "Data not available",
  "data" :
  {
   "method" : "VehicleInfo.GetDTCs"
  }
 }
}
```

GetVehicleData

Type

Function

Sender

SDL

Purpose

Get current values of specified vehicle data types.

NOTE

CLOUDAPPVEHICLEID

- An optional parameter used by cloud apps or the policy server to identify the head unit
- Could be used by a cloud app to identify an incoming connection from core
- Could be used by a policy server to index cloud app configurations for a specific head unit

The HMI will have to update this field if the user chooses to reset this value (in case the vehicle changes owners)

REQUEST

PARAMETERS

NAME	TYPE	MANDATORY	ADDITIONAL
gps	Boolean	false	
speed	Boolean	false	
rpm	Boolean	false	
fuelLevel	Boolean	false	
fuelLevel_State	Boolean	false	
instantFuelConsumption	Boolean	false	
externalTemperature	Boolean	false	
vin	Boolean	false	
prndl	Boolean	false	
tirePressure	Boolean	false	
odometer	Boolean	false	
beltStatus	Boolean	false	
bodyInformation	Boolean	false	
deviceStatus	Boolean	false	
driverBraking	Boolean	false	
wiperStatus	Boolean	false	
headLampStatus	Boolean	false	
engineTorque	Boolean	false	

NAME	TYPE	MANDATORY	ADDITIONAL
accPedalPosition	Boolean	false	
steeringWheelAngle	Boolean	false	
eCallInfo	Boolean	false	
airbagStatus	Boolean	false	
emergencyEvent	Boolean	false	
clusterModeStatus	Boolean	false	
myKey	Boolean	false	
turnSignal	Boolean	false	
fuelRange	Boolean	false	
engineOilLife	Boolean	false	
electronicParkBrakeStatus	Boolean	false	
cloudAppVehicleID	Boolean	false	

RESPONSE

PARAMETERS

NAME	TYPE	MANDATORY	ADDITIONAL
gps	Common.GPSData	false	
speed	Float	false	minvalue: 0 maxvalue: 700
rpm	Integer	false	minvalue: 0 maxvalue: 20000
fuelLevel	Float	false	minvalue: -6 maxvalue: 106
fuelLevel_State	Common.ComponentVolumeStatus	false	
instantFuelConsumption	Float	false	minvalue: 0 maxvalue: 25575
externalTemperature	Float	false	minvalue: -40 maxvalue: 100
vin	String	false	maxlength: 17
prndl	Common.PRNDL	false	
tirePressure	Common.TireStatus	false	
odometer	Integer	false	minvalue: 0 maxvalue: 17000000
beltStatus	Common.BeltStatus	false	
bodyInformation	Common.BodyInformation	false	
deviceStatus	Common.DeviceStatus	false	
driverBraking	Common.VehicleDataEventStatus	false	
wiperStatus	Common.WiperStatus	false	

NAME	TYPE	MANDATORY	ADDITIONAL
headLampStatus	Common.HeadLampStatus	false	
engineTorque	Float	false	minvalue: -1000 maxvalue: 2000
accPedalPosition	Float	false	minvalue: 0 maxvalue: 100
steeringWheelAngle	Float	false	minvalue: -2000 maxvalue: 2000
eCallInfo	Common.ECallInfo	false	
airbagStatus	Common.AirbagStatus	false	
emergencyEvent	Common.EmergencyEvent	false	
clusterModeStatus	Common.ClusterModeStatus	false	
myKey	Common.MyKey	false	
turnSignal	Common.TurnSignal	false	
fuelRange	Common.FuelRange	false	minsize=0 maxsize=100 array=true
engineOilLife	Float	false	minvalue=0 maxvalue=100
electronicParkBrakeStatus	Common.ElectronicParkBrakeStatus	false	
cloudAppVehicleID	String	false	

Sequence Diagrams

SEQUENCE DIAGRAM

GetVehicleData

View Diagram

Example Request

```
{
 "id" : 139,
 "jsonrpc" : "2.0",
 "method" : "VehicleInfo.GetVehicleData",
 "params" :
 {
  "gps" : true,
  "speed" : true,
  "fuelLevel_State" : true,
  "externalTemperature" : true,
  "prndl" : true,
  "tirePressure" : true,
  "odometer" : true,
  "beltStatus" : true,
  "bodyInformation" : true,
  "deviceStatus" : true,
  "wiperStatus" : true,
  "headLampStatus" : true,
  "accPedalPosition" : true
 }
}
```

Example Response

```
{
 "id" : 139,
 "jsonrpc" : "2.0",
 "result" :
 {
  "gps" :
  {
   "longitudeDegrees" : 46.4774700,
   "latitudeDegrees" : 30.7326200,
   "utcYear" : 2013,
   "utcMonth" : 12,
   "utcDay" : 31,
   "utcHours" : 23,
   "utcMinutes" : 50,
   "utcSeconds" : 5,
   "compassDirection" : "NORTH",
   "pdop" : 0.15,
   "hdop" : 1.01,
   "vdop" : 1.56,
   "actual" : true,
   "satellites" : 8,
   "dimension" : "3D",
   "altitude" : 47,
   "heading" : 0,
   "speed" : 90
  },
  "speed" : 90,
  "fuelLevel_State" : "LOW",
  "externalTemperature" : -5,
  "prndl" : "FOURTH",
  "tirePressure" :
  {
     "pressureTelltale" : "ON",
     "leftFront" : {
      "status" : "NORMAL"
     },
     "rightFront" : {
      "status" : "NORMAL"
     },
     "leftRear" : {
      "status" : "LOW"
     },
     "rightRear" : {
      "status" : "UNKNOWN"
     }
  },
  "odometer" : 1065,
  "beltStatus" :
  {
```

```
        "driverBeltDeployed" : "YES",
        "passengerBeltDeployed" : "YES",
      },
      "bodyInformation" :
      {
        "parkBrakeActive" : false,
        "ignitionStableStatus" : "IGNITION_SWITCH_STABLE",
        "ignitionStatus" : "RUN"
      },
      "deviceStatus" :
      {
        "voiceRecOn" : false,
        "btIconOn" : false,
        "callActive" : false,
        "phoneRoaming" : false,
        "textMsgAvailable" : true,
        "battLevelStatus" : "THREE_LEVEL_BARS",
        "stereoAudioOutputMuted" : true,
        "monoAudioOutputMuted" : false,
        "signalLevelStatus" : "NOT_PROVIDED",
        "primaryAudioSource" : "MOBILE_APP",
        "eCallEventActive" : false
      },
      "wiperStatus" : "OFF",
      "headLampStatus" :
      {
        "lowBeamsOn" : true,
        "highBeamsOn" : false
      },
      "accPedalPosition" : 80,
      "code" : 0,
      "method" : "VehicleInfo.GetVehicleData"
    }
  }
```

Example Error


```
{
 "id" : 139,
 "jsonrpc" : "2.0",
 "error" :
 {
  "code" : 9,
  "message" : "The requested data is not available",
  "data" :
  {
   "method" : "VehicleInfo.GetVehicleData"
  }
 }
}
```


GetVehicleType

Type Function
Sender SDL
Purpose Get general information about the vehicle.

REQUEST

PARAMETERS

NAME	TYPE	MANDATORY	ADDITIONAL

RESPONSE

PARAMETERS

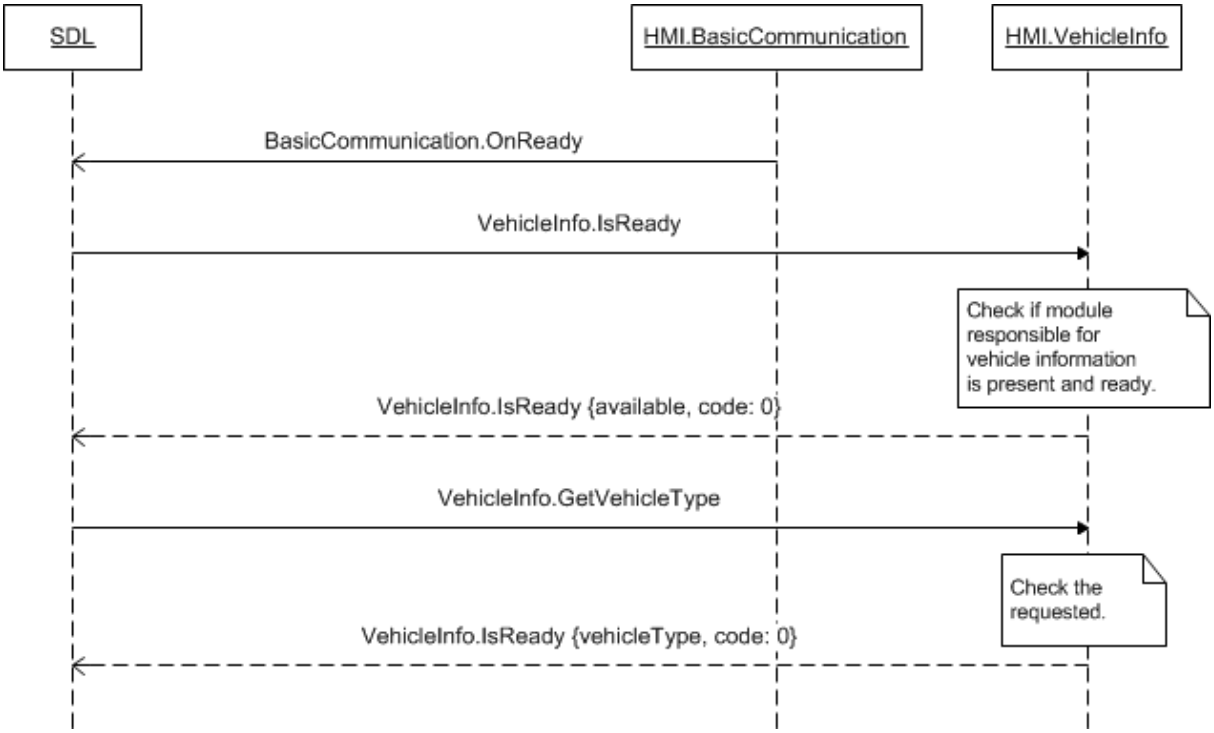
NAME	TYPE	MANDATORY	ADDITIONAL
vehicleType	Common.VehicleType	true	

Sequence Diagrams

SEQUENCE DIAGRAM

GetVehicleType

View Diagram



Example Request

```
{
  "id" : 21,
  "jsonrpc" : "2.0",
  "method" : "VehicleInfo.GetVehicleType"
}
```

Example Response


```
{
 "id" : 21,
 "jsonrpc" : "2.0",
 "result" :
 {
  "vehicleType" :
  [
   "make" : "Ford",
   "model" : "Fusion",
   "modelYear" : "2013",
   "trim" : "SE"
  ]
  "code" : 0,
  "method" : "VehicleInfo.GetVehicleType"
 }
}
```


Example Error


```
{
 "id" : 21,
 "jsonrpc" : "2.0",
 "error" :
 {
  "code" : 9,
  "message" : "The requested data is not available",
  "data" :
  {
   "method" : "VehicleInfo.GetVehicleType"
  }
 }
}
```

IsReady

Type

Function

Sender

SDL

Purpose

Request ready state of the VehicleInfo Module

REQUEST

NOTE

1. SDL sends `VehicleInfo.IsReady` request after HMI confirms its readiness via `BC.OnReady` notification.
2. If HMI responds with `"available":false`, SDL will not further communicate over VehicleInfo interface with HMI.
3. If HMI does not respond during SDL's default timeout, SDL will continue to send RPCs over VehicleInfo interface to HMI.

PARAMETERS

The request does not have specific parameters.

RESPONSE

MUST

1. Check whether VehicleInfo module is available and ready.
2. Respond correspondingly to results of this check.

PARAMETERS

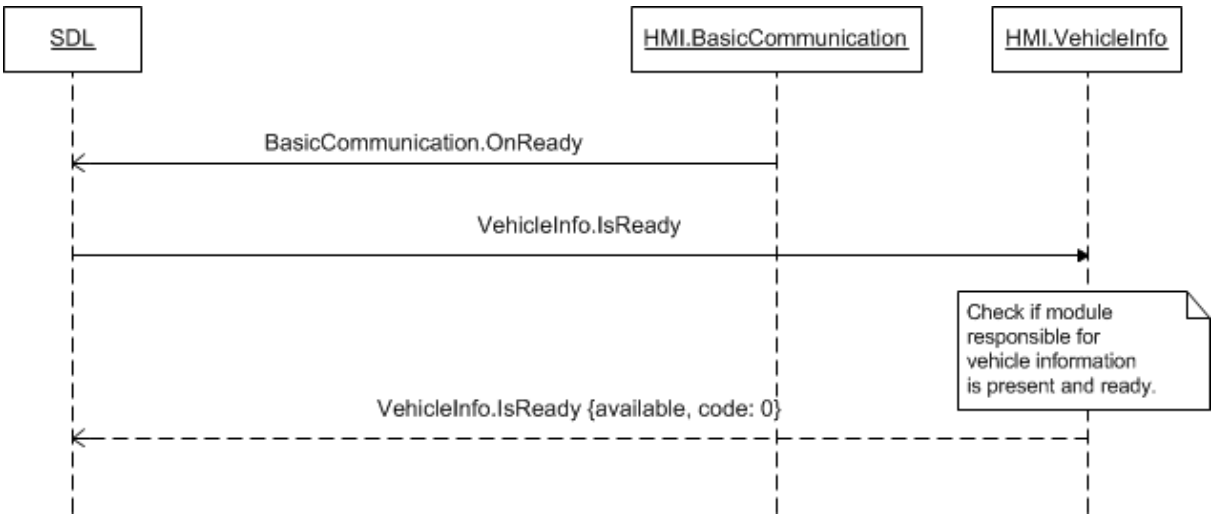
NAME	TYPE	MANDATORY	ADDITIONAL
available	Boolean	true	

Sequence Diagrams

SEQUENCE DIAGRAM

IsReady

View Diagram



Example Request

```
{
 "id" : 17,
 "jsonrpc" : "2.0",
 "method" : "VehicleInfo.IsReady"
}
```

Example Response

```
{
 "id" : 17,
 "jsonrpc" : "2.0",
 "result" :
 {
  "availabe" : true,
  "code" : 0,
  "method" : "VehicleInfo.IsReady"
 }
}
```

Example Error

```
{
 "id" : 17,
 "jsonrpc" : "2.0",
 "error" :
 {
  "code" : 9,
  "message" : "Data not available",
  "data" :
  {
   "method" : "VehicleInfo.IsReady"
  }
 }
}
```

OnVehicleData

Type Notification
Sender HMI
Purpose Inform SDL about changes to subscribed vehicle data values.

NOTE

CLOUDAPPVEHICLEID

- An optional parameter used by cloud apps or the policy server to identify the head unit
- Could be used by a cloud app to identify an incoming connection from core
- Could be used by a policy server to index cloud app configurations for a specific head unit

The HMI will have to update this field if the user chooses to reset this value (in case the vehicle changes owners)

Notification

PARAMETERS

NAME	TYPE	MANDATORY	ADDITIONAL
gps	Common.GPSData	false	
speed	Float	false	minvalue: 0 maxvalue: 700
rpm	Integer	false	minvalue: 0 maxvalue: 20000
fuelLevel	Float	false	minvalue: -6 maxvalue: 106
fuelLevel_State	Common.ComponentVolumeStatus	false	
instantFuelConsumption	Float	false	minvalue: 0 maxvalue: 25575
externalTemperature	Float	false	minvalue: -40 maxvalue: 100
vin	String	false	maxlength: 17
prndl	Common.PRNDL	false	
tirePressure	Common.TireStatus	false	
odometer	Integer	false	minvalue: 0 maxvalue: 17000000
beltStatus	Common.BeltStatus	false	
bodyInformation	Common.BodyInformation	false	
deviceStatus	Common.DeviceStatus	false	
driverBraking	Common.VehicleDataEventStatus	false	
wiperStatus	Common.WiperStatus	false	

NAME	TYPE	MANDATORY	ADDITIONAL
headLampStatus	Common.HeadLampStatus	false	
engineTorque	Float	false	minvalue: -1000 maxvalue: 2000
accPedalPosition	Float	false	minvalue: 0 maxvalue: 100
steeringWheelAngle	Float	false	minvalue: -2000 maxvalue: 2000
eCallInfo	Common.ECallInfo	false	
airbagStatus	Common.AirbagStatus	false	
emergencyEvent	Common.EmergencyEvent	false	
clusterModeStatus	Common.ClusterModeStatus	false	
myKey	Common.MyKey	false	
turnSignal	Common.TurnSignal	false	
fuelRange	Common.FuelRange	false	minsize=0 maxsize=100 array=true
engineOilLife	Float	false	minvalue=0 maxvalue=100
electronicParkBrakeStatus	Common.ElectronicParkBrakeStatus	false	
cloudAppVehicleID	String	false	

Sequence Diagrams

SEQUENCE DIAGRAM

OnVehicleData

View Diagram

JSON EXAMPLE NOTIFICATION

```
{
  "jsonrpc" : "2.0",
  "method" : "VehicleInfo.OnVehicleData",
  "params" :
  {
    "speed" : 60,
    "externalTemperature" : -7,
    "prndl" : "THIRD",
    "odometer" : 1066,
    "wiperStatus" : "MAN_INT_ON",
    "accPedalPosition" : 70
  }
}
```

ReadDID

Type Function
Sender

SDL

Purpose

Request data from specified DID locations.

REQUEST

PARAMETERS

NAME	TYPE	MANDATORY	ADDITIONAL
ecuName	Integer	true	minvalue: 0 maxvalue: 65535 array: true minsize: 1 maxsize: 1000
didLocation	Integer	true	minvalue: 0 maxvalue: 65535
appID	Integer	true	

RESPONSE

PARAMETERS

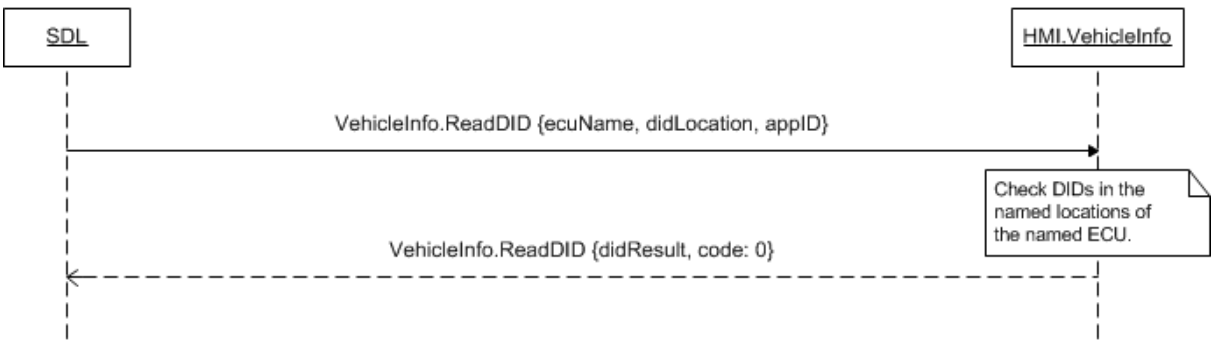
NAME	TYPE	MANDATORY	ADDITIONAL
didResult	Common.DIDResult	false	array: true minsize: 0 maxsize: 1000

Sequence Diagrams

SEQUENCE DIAGRAM

ReadDID General Processing

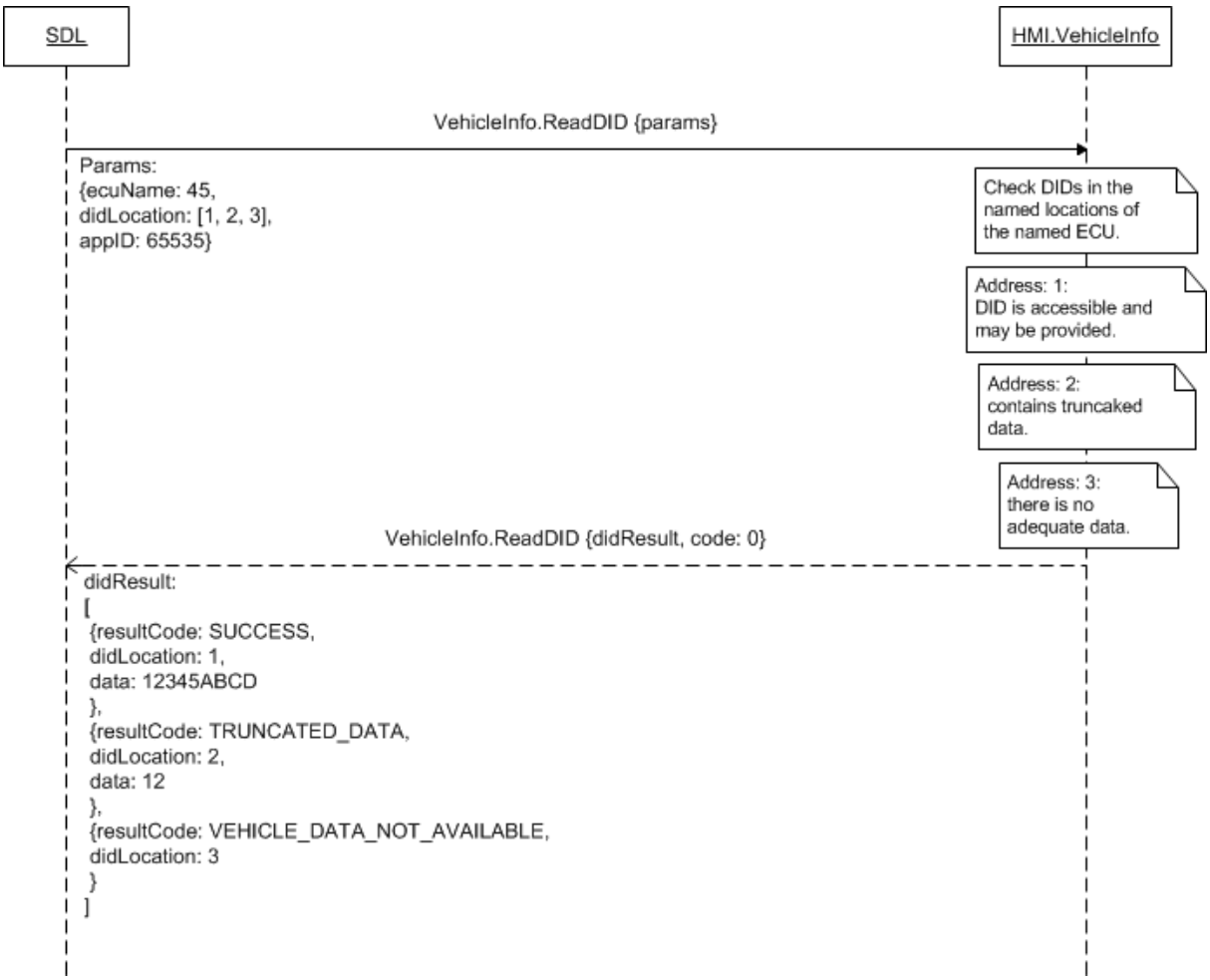
View Diagram



SEQUENCE DIAGRAM

ReadDID Expanded result

View Diagram



Example Request


```
{
 "id" : 158,
 "jsonrpc" : "2.0",
 "method" : "VehicleInfo.ReadDID",
 "params" :
 {
  "ecuName" : 1287,
  "didLocation" : [35, 48, 182],
  "appID" : 93
 }
}
```

Example Response


```
{
 "id" : 158,
 "jsonrpc" : "2.0",
 "result" :
 {
  "didResult" :
  [
   {
   "resultCode" : SUCCESS,
   "didLocation" : 35,
   "data" : "38AF"
   },
   {
   "resultCode" : TRUNCATED_DATA,
   "didLocation" : 48,
   "data" : "35"
   },
   {
   "resultCode" : INVALID_ID,
   "didLocation" : 182
   }
  ],
  "code" : 0,
  "method" : "VehicleInfo.ReadDID"
 }
}
```

Example Error


```
{
 "id" : 158,
 "jsonrpc" : "2.0",
 "error" :
 {
  "code" : 2,
  "message" : "The requested ECU does not exist",
  "data" :
  {
   "method" : "VehicleInfo.ReadDID"
  }
 }
}
```


SubscribeVehicleData

Type Function
Sender SDL
Purpose Subscribe to periodic vehicle data type updates

REQUEST

PARAMETERS

NAME	TYPE	MANDATORY	ADDITIONAL
gps	Boolean	false	
speed	Boolean	false	
rpm	Boolean	false	
fuelLevel	Boolean	false	
fuelLevel_State	Boolean	false	
instantFuelConsumption	Boolean	false	
externalTemperature	Boolean	false	
prndl	Boolean	false	
tirePressure	Boolean	false	
odometer	Boolean	false	
beltStatus	Boolean	false	
bodyInformation	Boolean	false	
deviceStatus	Boolean	false	
driverBraking	Boolean	false	
wiperStatus	Boolean	false	
headLampStatus	Boolean	false	
engineTorque	Boolean	false	
accPedalPosition	Boolean	false	

NAME	TYPE	MANDATORY	ADDITIONAL
steeringWheelAngle	Boolean	false	
eCallInfo	Boolean	false	
airbagStatus	Boolean	false	
emergencyEvent	Boolean	false	
clusterModeStatus	Boolean	false	
myKey	Boolean	false	
turnSignal	Boolean	false	
fuelRange	Boolean	false	
engineOilLife	Boolean	false	
electronicParkBrakeStatus	Boolean	false	
cloudAppVehicleID	Boolean	false	

RESPONSE

PARAMETERS

NAME	TYPE	MANDATORY	ADDITIONAL
gps	Common.VehicleDataResult	false	
speed	Common.VehicleDataResult	false	
rpm	Common.VehicleDataResult	false	
fuelLevel	Common.VehicleDataResult	false	
fuelLevel_State	Common.VehicleDataResult	false	
instantFuelConsumption	Common.VehicleDataResult	false	
externalTemperature	Common.VehicleDataResult	false	
prndl	Common.VehicleDataResult	false	
tirePressure	Common.VehicleDataResult	false	
odometer	Common.VehicleDataResult	false	
beltStatus	Common.VehicleDataResult	false	
bodyInformation	Common.VehicleDataResult	false	
deviceStatus	Common.VehicleDataResult	false	
driverBraking	Common.VehicleDataResult	false	
wiperStatus	Common.VehicleDataResult	false	
headLampStatus	Common.VehicleDataResult	false	
engineTorque	Common.VehicleDataResult	false	
accPedalPosition	Common.VehicleDataResult	false	

NAME	TYPE	MANDATORY	ADDITIONAL
steeringWheelAngle	Common.VehicleDataResult	false	
eCallInfo	Common.VehicleDataResult	false	
airbagStatus	Common.VehicleDataResult	false	
emergencyEvent	Common.VehicleDataResult	false	
clusterModes	Common.VehicleDataResult	false	
myKey	Common.VehicleDataResult	false	
turnSignal	Common.VehicleDataResult	false	
fuelRange	Common.VehicleDataResult	false	
engineOilLife	Common.VehicleDataResult	false	
electronicParkBrakeStatus	Common.VehicleDataResult	false	
cloudAppVehicleID	Common.VehicleDataResult	false	

Sequence Diagrams

SEQUENCE DIAGRAM

SubscribeVehicleData

View Diagram

Example Request


```
{
  "id" : 139,
  "jsonrpc" : "2.0",
  "method" : "VehicleInfo.SubscribeVehicleData",
  "params" :
  {
    "gps" : true,
    "speed" : true,
    "fuelLevel_State" : true,
    "externalTemperature" : true,
    "prndl" : true,
    "tirePressure" : true,
    "odometer" : true,
    "beltStatus" : true,
    "bodyInformation" : true,
    "deviceStatus" : true,
    "wiperStatus" : true,
    "headLampStatus" : true,
    "accPedalPosition" : true
  }
}
```

Example Response

```
{
 "id" : 139,
 "jsonrpc" : "2.0",
 "result" :
 {
  "gps" :
  {
   "dataType" : "VEHICLEDATA_GPS",
   "resultCode" : "SUCCESS"
  },

  "speed" :
  {
   "dataType" : "VEHICLEDATA_SPEED",
   "resultCode" : "DATA_ALREADY_SUBSCRIBED"
  },

  "fuelLevel_State" :
  {
   "dataType" : "VEHICLEDATA_FUELLEVEL",
   "resultCode" : "SUCCESS"
  },

  "externalTemperature" :
  {
   "dataType" : "VEHICLEDATA_EXTERNTEMP",
   "resultCode" : "VEHICLE_DATA_NOT_AVAILABLE"
  },

  "prndl" :
  {
   "dataType" : "VEHICLEDATA_PRNDL",
   "resultCode" : "VEHICLE_DATA_NOT_AVAILABLE"
  },

  "tirePressure" :
  {
   "dataType" : "VEHICLEDATA_TIREPRESSURE",
   "resultCode" : "SUCCESS"
  },

  "odometer" :
  {
   "dataType" : "VEHICLEDATA_ODOMETER",
   "resultCode" : "SUCCESS"
  },

  "beltStatus" :
  {
```

```
      "dataType" : "VEHICLEDATA_BELTSTATUS",
      "resultCode" : "SUCCESS"
    },

    "bodyInformation" :
    {
      "dataType" : "VEHICLEDATA_BODYINFO",
      "resultCode" : "SUCCESS"
    },

    "deviceStatus" :
    {
      "dataType" : "VEHICLEDATA_DEVICESTATUS",
      "resultCode" : "DATA_ALREADY_SUBSCRIBED"
    },

    "wiperStatus" :
    {
      "dataType" : "VEHICLEDATA_WIPERSTATUS",
      "resultCode" : "SUCCESS"
    },

    "headLampStatus" :
    {
      "dataType" : "VEHICLEDATA_HEADLAMPSTATUS",
      "resultCode" : "SUCCESS"
    },

    "accPedalPosition" :
    {
      "dataType" : "VEHICLEDATA_ACCPEDAL",
      "resultCode" : "VEHICLE_DATA_NOT_AVAILABLE"
    },

    "code" : 0,
    "method" : "VehicleInfo.SubscribeVehicleData"
  }
}
```

Example Error


```
{
 "id" : 139,
 "jsonrpc" : "2.0",
 "error" :
 {
  "code" : 6,
  "message" : "All of requested data types is subscribed already",
  "data" :
  {
   "method" : "VehicleInfo.SubscribeVehicleData"
  }
 }
}
```


UnsubscribeVehicleData

Type Function
Sender SDL
Purpose Unsubscribe from periodic vehicle data type updates

REQUEST

PARAMETERS

NAME	TYPE	MANDATORY	ADDITIONAL
gps	Boolean	false	
speed	Boolean	false	
rpm	Boolean	false	
fuelLevel	Boolean	false	
fuelLevel_State	Boolean	false	
instantFuelConsumption	Boolean	false	
externalTemperature	Boolean	false	
prndl	Boolean	false	
tirePressure	Boolean	false	
odometer	Boolean	false	
beltStatus	Boolean	false	
bodyInformation	Boolean	false	
deviceStatus	Boolean	false	
driverBraking	Boolean	false	
wiperStatus	Boolean	false	
headLampStatus	Boolean	false	
engineTorque	Boolean	false	
accPedalPosition	Boolean	false	

NAME	TYPE	MANDATORY	ADDITIONAL
steeringWheelAngle	Boolean	false	
eCallInfo	Boolean	false	
airbagStatus	Boolean	false	
emergencyEvent	Boolean	false	
clusterModeStatus	Boolean	false	
myKey	Boolean	false	
turnSignal	Boolean	false	
fuelRange	Boolean	false	
engineOilLife	Boolean	false	
electronicParkBrakeStatus	Boolean	false	
cloudAppVehicleID	Boolean	false	

RESPONSE

PARAMETERS

NAME	TYPE	MANDATORY	ADDITIONAL
gps	Common.VehicleDataResult	false	
speed	Common.VehicleDataResult	false	
rpm	Common.VehicleDataResult	false	
fuelLevel	Common.VehicleDataResult	false	
fuelLevel_State	Common.VehicleDataResult	false	
instantFuelConsumption	Common.VehicleDataResult	false	
externalTemperature	Common.VehicleDataResult	false	
prndl	Common.VehicleDataResult	false	
tirePressure	Common.VehicleDataResult	false	
odometer	Common.VehicleDataResult	false	
beltStatus	Common.VehicleDataResult	false	
bodyInformation	Common.VehicleDataResult	false	
deviceStatus	Common.VehicleDataResult	false	
driverBraking	Common.VehicleDataResult	false	
wiperStatus	Common.VehicleDataResult	false	
headLampStatus	Common.VehicleDataResult	false	
engineTorque	Common.VehicleDataResult	false	
accPedalPosition	Common.VehicleDataResult	false	

NAME	TYPE	MANDATORY	ADDITIONAL
steeringWheelAngle	Common.VehicleDataResult	false	
eCallInfo	Common.VehicleDataResult	false	
airbagStatus	Common.VehicleDataResult	false	
emergencyEvent	Common.VehicleDataResult	false	
clusterModes	Common.VehicleDataResult	false	
myKey	Common.VehicleDataResult	false	
turnSignal	Common.VehicleDataResult	false	
fuelRange	Common.VehicleDataResult	false	
engineOilLife	Common.VehicleDataResult	false	
electronicParkBrakeStatus	Common.VehicleDataResult	false	
cloudAppVehicleID	Common.VehicleDataResult	false	

Sequence Diagrams

SEQUENCE DIAGRAM

UnsubscribeVehicleData

View Diagram

SEQUENCE DIAGRAM

UnsubscribeVehicleData unexpected disconnect

View Diagram

Example Request

```
{
 "id" : 139,
 "jsonrpc" : "2.0",
 "method" : "VehicleInfo.UnsubscribeVehicleData",
 "params" :
 {
  "gps" : true,
  "speed" : true,
  "fuelLevel_State" : true,
  "externalTemperature" : true,
  "prndl" : true,
  "tirePressure" : true,
  "odometer" : true,
  "beltStatus" : true,
  "bodyInformation" : true,
  "deviceStatus" : true,
  "wiperStatus" : true,
  "headLampStatus" : true,
  "accPedalPosition" : true
 }
}
```

Example Response

```
"id" : 139,
"jsonrpc" : "2.0",
"result" :
{
 "gps" :
 {
  "dataType" : "VEHICLEDATA_GPS",
  "resultCode" : "SUCCESS"
 },

 "speed" :
 {
  "dataType" : "VEHICLEDATA_SPEED",
  "resultCode" : "DATA_NOT_SUBSCRIBED"
 },

 "fuelLevel_State" :
 {
  "dataType" : "VEHICLEDATA_FUELLEVEL",
  "resultCode" : "SUCCESS"
 },

 "externalTemperature" :
 {
  "dataType" : "VEHICLEDATA_EXTERNTEMP",
  "resultCode" : "DATA_NOT_SUBSCRIBED"
 },

 "prndl" :
 {
  "dataType" : "VEHICLEDATA_PRNDL",
  "resultCode" : "DATA_NOT_SUBSCRIBED"
 },

 "tirePressure" :
 {
  "dataType" : "VEHICLEDATA_TIREPRESSURE",
  "resultCode" : "SUCCESS"
 },

 "odometer" :
 {
  "dataType" : "VEHICLEDATA_ODOMETER",
  "resultCode" : "SUCCESS"
 },

 "beltStatus" :
 {
  "dataType" : "VEHICLEDATA_BELTSTATUS",
```

```
      "resultCode" : "SUCCESS"
    },

    "bodyInformation" :
    {
      "dataType" : "VEHICLEDATA_BODYINFO",
      "resultCode" : "SUCCESS"
    },

    "deviceStatus" :
    {
      "dataType" : "VEHICLEDATA_DEVICESTATUS",
      "resultCode" : "DATA_NOT_SUBSCRIBED"
    },

    "wiperStatus" :
    {
      "dataType" : "VEHICLEDATA_WIPERSTATUS",
      "resultCode" : "SUCCESS"
    },

    "headLampStatus" :
    {
      "dataType" : "VEHICLEDATA_HEADLAMPSTATUS",
      "resultCode" : "SUCCESS"
    },

    "accPedalPosition" :
    {
      "dataType" : "VEHICLEDATA_ACCPEDAL",
      "resultCode" : "DATA_NOT_SUBSCRIBED"
    },

    "code" : 0,
    "method" : "VehicleInfo.UnsubscribeVehicleData"
  }
}
```

Example Error


```
{
 "id" : 139,
 "jsonrpc" : "2.0",
 "error" :
 {
  "code" : 22,
  "message" : "An unknown error occurred",
  "data" :
  {
   "method" : "VehicleInfo.UnsubscribeVehicleData"
  }
 }
}
```


AddCommand

Type Function
Sender SDL
Purpose Add a command for voice recognition

REQUEST

If the application sends `AddCommand` with the `vrCommands` parameter then SDL must maintain a list of the added `vrCommands`.
For each AddCommand, only the first item in the `vrCommands` array shall be added to the list.

Whenever the internal list of added `vrCommands` is updated SDL must:
construct the `vrHelp` and `helpPrompt` parameters using the data from the list SDL created internally
send these parameters to the HMI via the `SetGlobalProperties` RPC

PARAMETERS

NAME	TYPE	MANDATORY	ADDITIONAL
cmdID	Integer	true	minvalue: 0 maxvalue: 2000000000
vrCommands	String	true	array: true minsize: 1 maxsize: 100 maxlength: 99
type	Common.VRCommandType	true	
grammarID	Integer	true	minvalue: 0 maxvalue: 2000000000
appID	Integer	false	

RESPONSE

PARAMETERS

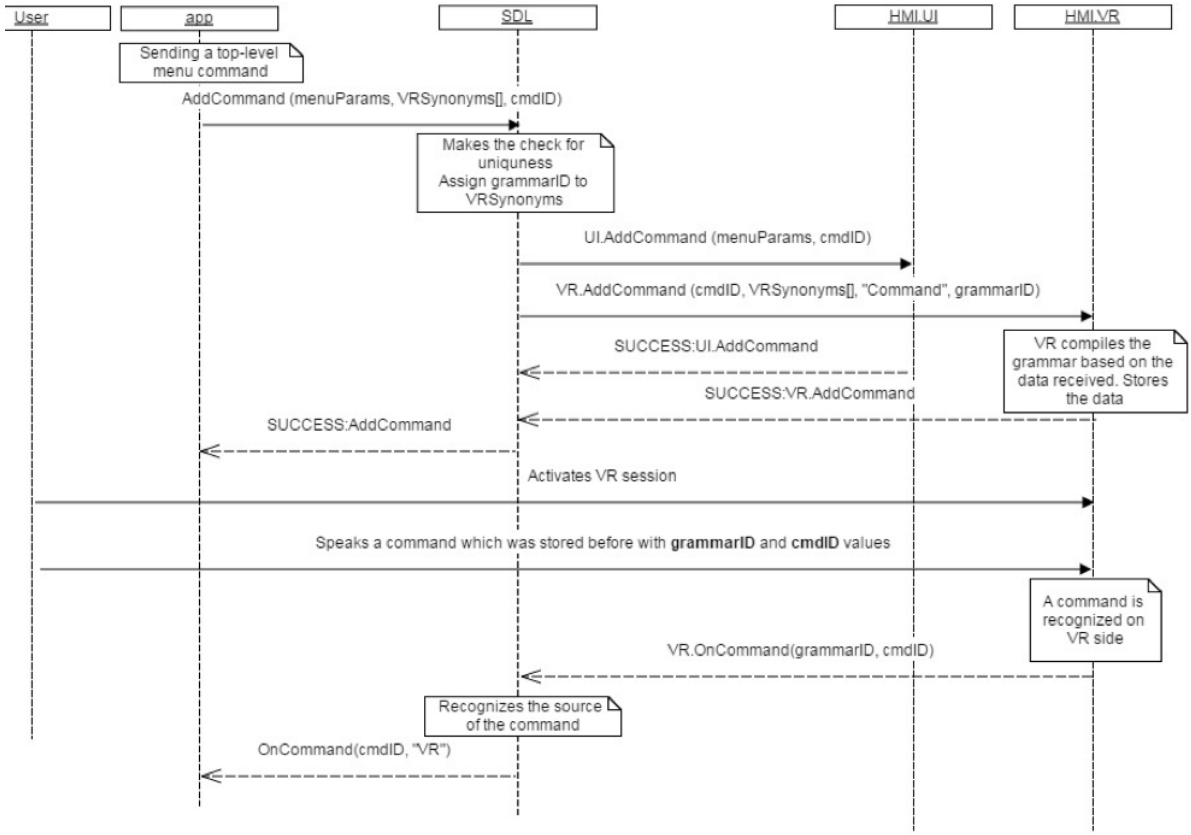
This RPC has no additional parameter requirements

Sequence Diagrams

SEQUENCE DIAGRAM

AddCommand

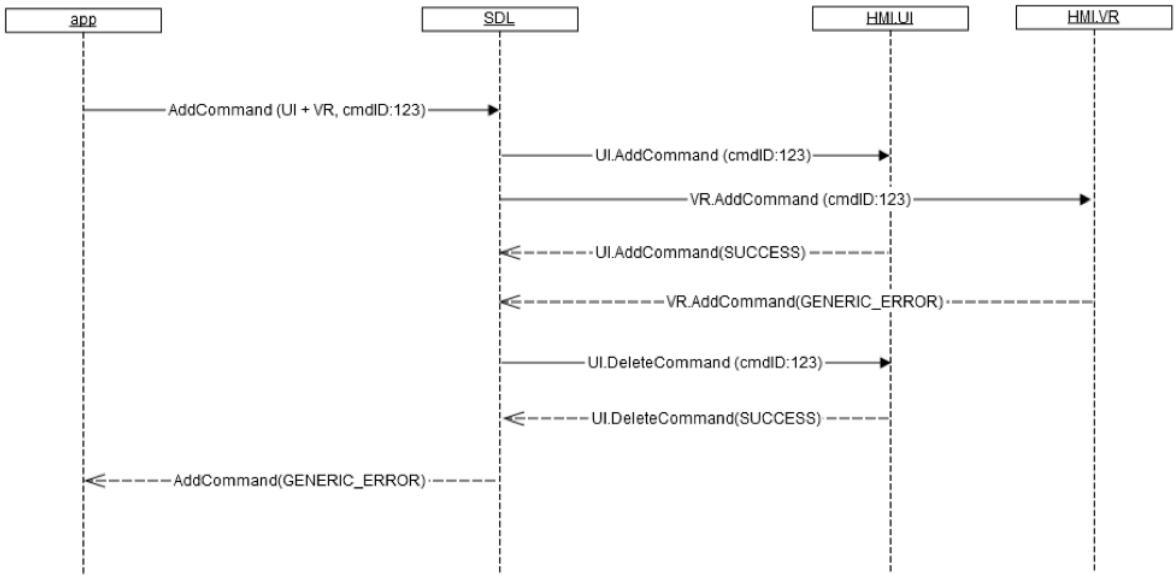
View Diagram



SEQUENCE DIAGRAM

UI.AddCommand returns SUCCESS, VR.AddCommand fails

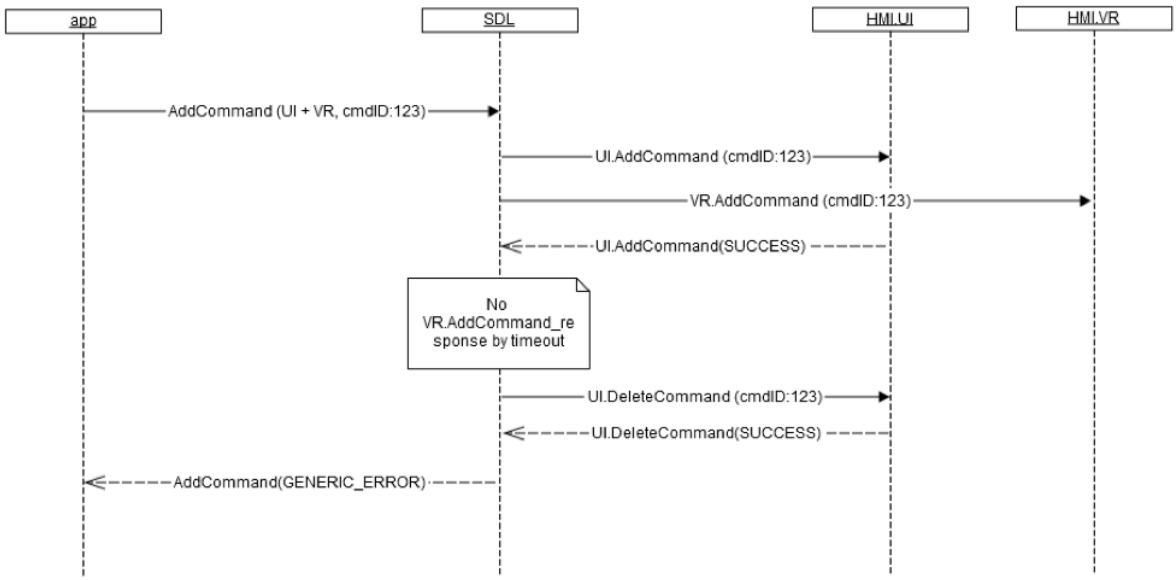
View Diagram



SEQUENCE DIAGRAM

UI.AddCommand returns SUCCESS, VR.AddCommand no response

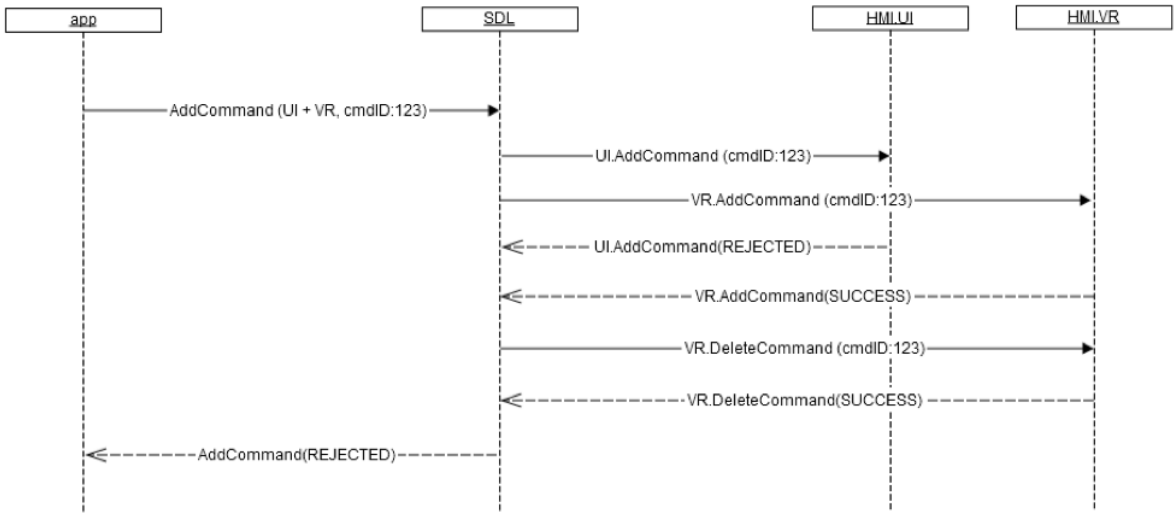
View Diagram



SEQUENCE DIAGRAM

UI.AddCommand fails, VR.AddCommand returns SUCCESS

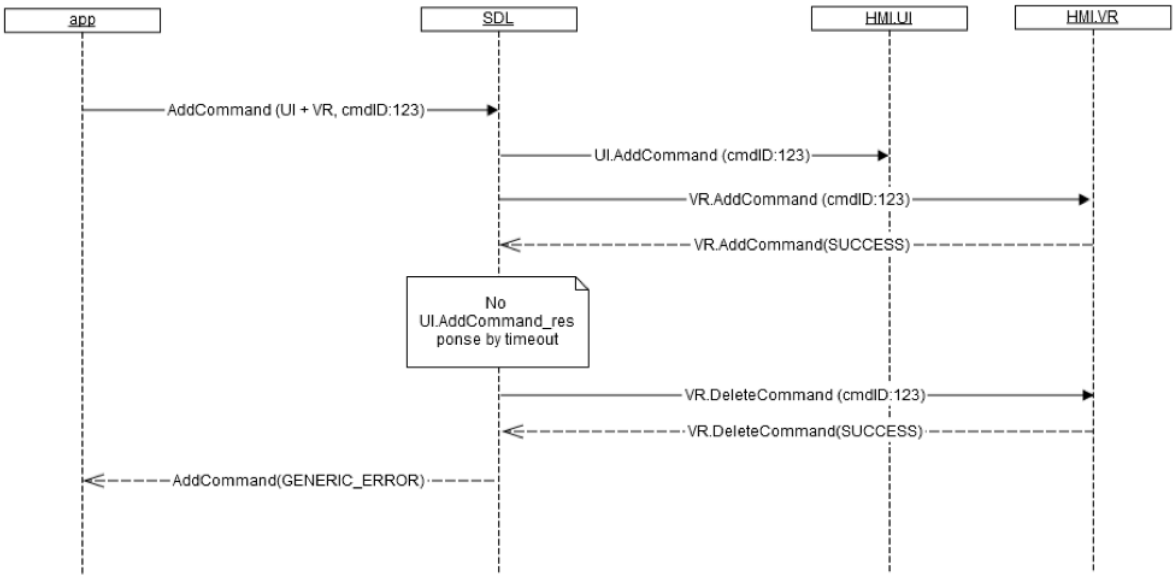
View Diagram



SEQUENCE DIAGRAM

UI.AddCommand no response, VR.AddCommand returns SUCCESS

View Diagram



Example Request

```
{
 "id" : 119,
 "jsonrpc" : "2.0",
 "method" : "VR. AddCommand",
 "params" :
 {
  "cmdID" : 4365,
  "vrCommands" :
  [
      "Leave",
      "Exit",
      "Quit"
  ],
   "grammarID":123,
 "appID" : 64467
 }
}
```

Example Response


```
{
 "id" : 119,
 "jsonrpc" : "2.0",
 "result" :
 {
  "code" : 0,
  "method" : "VR.AddCommand"
 }
}
```


Example Error


```
{
 "id" : 119,
 "jsonrpc" : "2.0",
 "error" :
 {
  "code" : 13,
  "message" : "Provided appID is not valid",
  "data" :
  {
   "method" : "VR.AddCommand"
  }
 }
}
```

ChangeRegistration

Type

Function

Sender

SDL

Purpose

Change the language for voice recognition.

REQUEST

PARAMETERS

NAME	TYPE	MANDATORY	ADDITIONAL
vrSynonyms	String	false	array: true minsize: 1 maxsize: 100 maxlength: 40
language	Common.Language	true	
appID	Integer	true	

RESPONSE

PARAMETERS

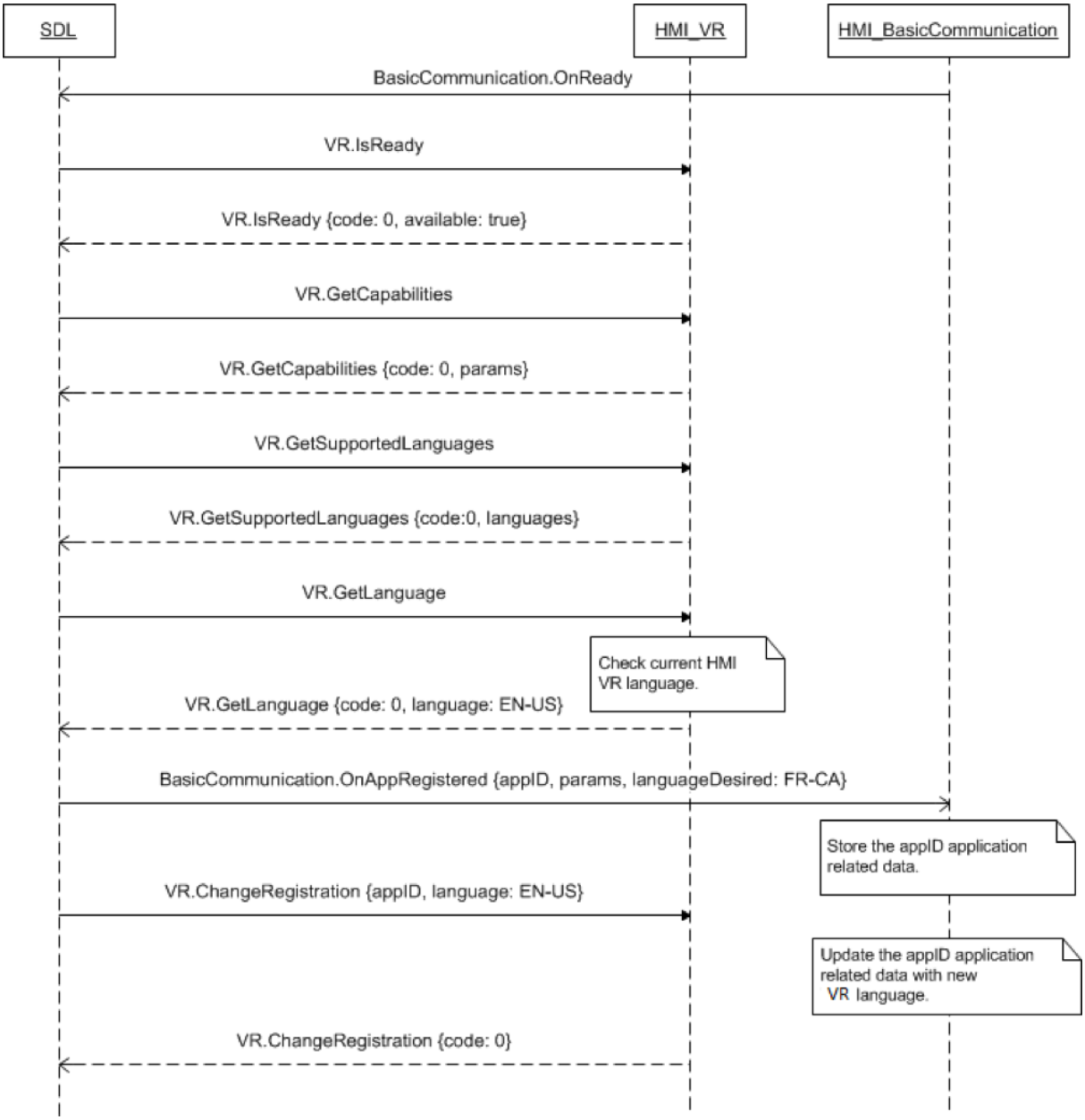
This RPC has no additional parameter requirements

Sequence Diagrams

SEQUENCE DIAGRAM

ChangeRegistration

View Diagram



Example Request

```
{
 "id" : 206,
 "jsonrpc" : "2.0",
 "method" : "VR.ChangeRegistration",
 "params" :
 {
  "language" : "DE-DE",
  "appID" : 13264
 }
}
```

Example Response


```
{
 "id" : 206,
 "jsonrpc" : "2.0",
 "result" :
 {
  "code" : 0,
  "method" : "VR.ChangeRegistration"
 }
}
```


Example Error


```
{
 "id" : 206,
 "jsonrpc" : "2.0",
 "error" :
 {
  "code" : 22,
  "message" : "The unknown error occurred",
  "data" :
  {
   "method" : "VR.ChangeRegistration"
  }
 }
}
```

DeleteCommand

Type Function
Sender SDL
Purpose

Remove a voice recognition command from the application's context.

REQUEST

PARAMETERS

NAME	TYPE	MANDATORY	ADDITIONAL
cmdID	Integer	true	minvalue: 0 maxvalue: 2000000000
type	Common.VRCommandType	true	
grammarID	Integer	true	minvalue: 0 maxvalue: 2000000000
appID	Integer	true	

RESPONSE

PARAMETERS

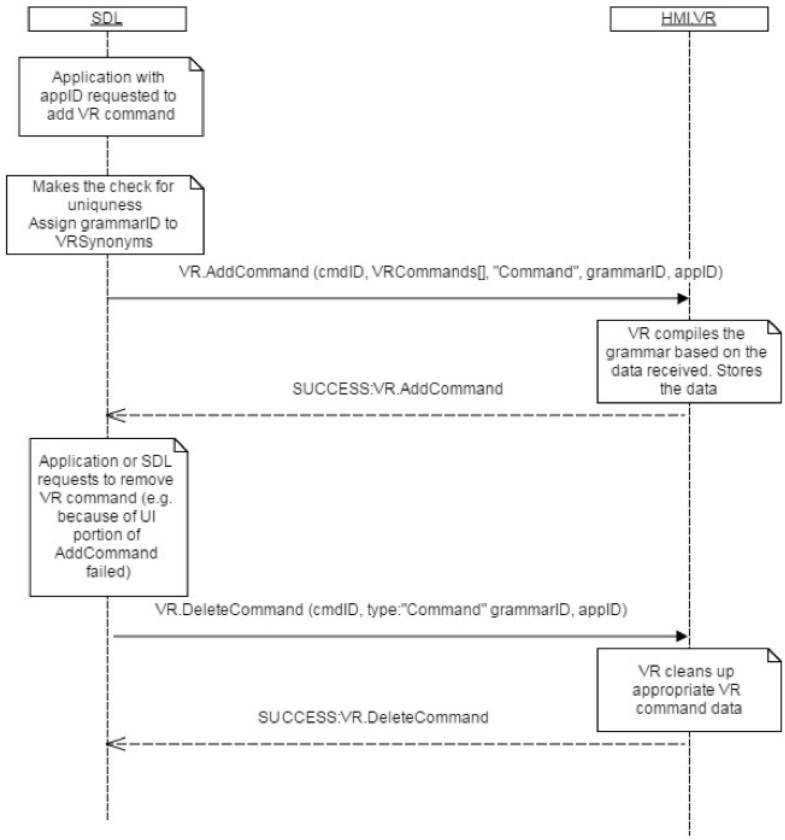
This RPC has no additional parameter requirements

Sequence Diagrams

SEQUENCE DIAGRAM

DeleteCommand

View Diagram



Example Request

```
{
 "id" : 147,
 "jsonrpc" : "2.0",
 "method" : "VR. DeleteCommand",
 "params" :
 {
  "cmdID" : 4365,
    "type":"Command",
    "grammarID":13,
  "appID" : 8764
 }
}
```

Example Response


```
{
 "id" : 147,
 "jsonrpc" : "2.0",
 "result" :
 {
  "code" : 0,
  "method" : "VR.DeleteCommand"
 }
}
```


Example Error


```
{
 "id" : 147,
 "jsonrpc" : "2.0",
 "error" :
 {
  "code" : 13,
  "message" : "One of the provided IDs is not valid",
  "data" :
  {
   "method" : "VR.DeleteCommand"
  }
 }
}
```

GetCapabilities

Type Function
Sender SDL
Purpose Inform SDL of the VR capabilities of the vehicle.

REQUEST

PARAMETERS

NAME	TYPE	MANDATORY	ADDITIONAL

RESPONSE

PARAMETERS

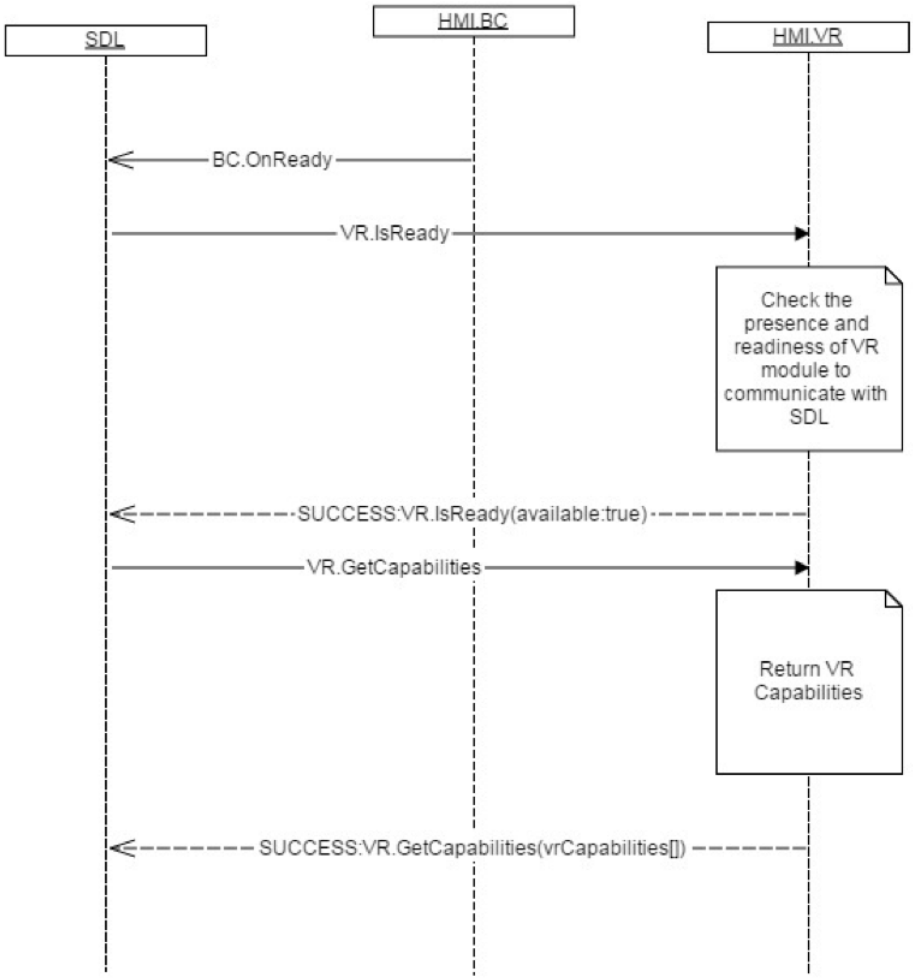
NAME	TYPE	MANDATORY	ADDITIONAL
vrCapabilities	Common.VrCapabilities	false	array: true minsize: 1 maxsize: 100

Sequence Diagrams

SEQUENCE DIAGRAM

GetCapabilities

View Diagram



Example Request

```
{
  "id" : 9,
  "jsonrpc" : "2.0",
  "method" : "VR.GetCapabilities"
}
```

Example Response

```
{
 "id" : 9,
 "jsonrpc" : "2.0",
 "result" :
 {
  "vrCapabilities" : [TEXT],
  "code" : 0,
  "method" : "VR.GetCapabilities"
 }
}
```

Example Error

```
{
 "id" : 9,
 "jsonrpc" : "2.0",
 "error" :
 {
  "code" : 11,
  "message" : "The data sent is invalid",
  "data" :
  {
   "method" : "VR.GetCapabilities"
  }
 }
}
```

GetLanguage

Type

Function

Sender

SDL

Purpose

Get the current voice recognition language.

REQUEST

PARAMETERS

NAME	TYPE	MANDATORY	ADDITIONAL

RESPONSE

PARAMETERS

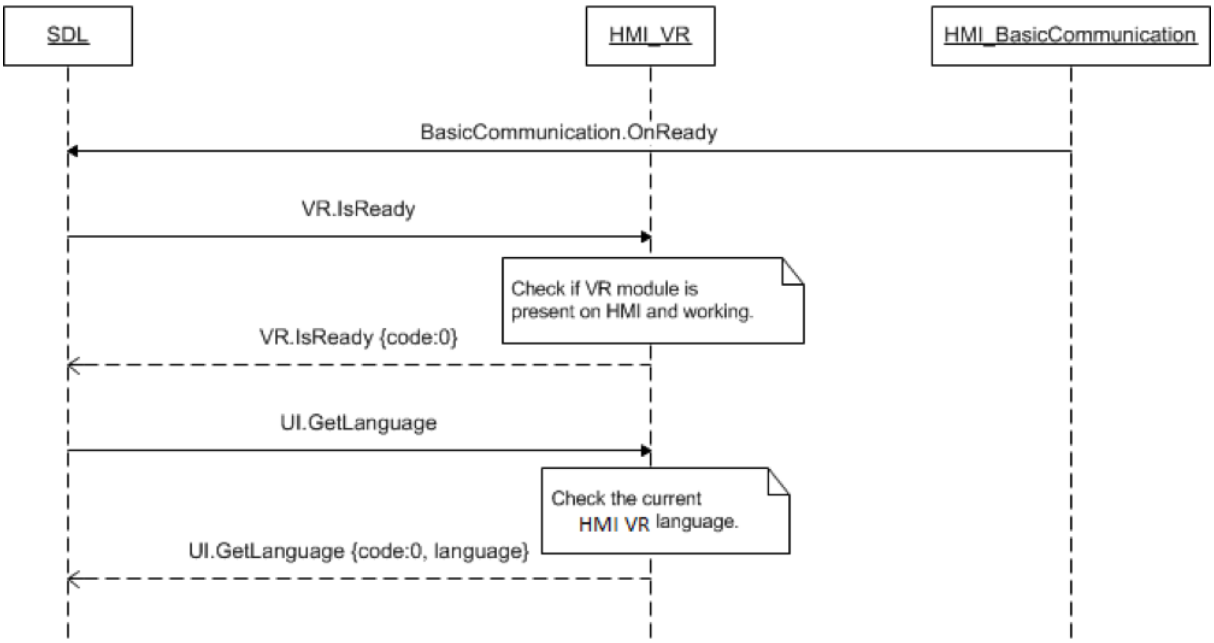
NAME	TYPE	MANDATORY	ADDITIONAL
language	Common.Language	true	

Sequence Diagrams

SEQUENCE DIAGRAM

GetLanguage

View Diagram



Example Request

```
{
 "id" : 110,
 "jsonrpc" : "2.0",
 "method" : "VR.GetLanguage",
}
```

Example Response

```
{
 "id" : 110,
 "jsonrpc" : "2.0",
 "result" :
 {
  "language" : "DE-DE",
  "code" : 0,
  "method" : "VR.GetLanguage"
 }
}
```

Example Error


```
{
 "id" : 110,
 "jsonrpc" : "2.0",
 "error" :
 {
  "code" : 22,
  "message" : "During the API call the unknown error has occured",
  "data" :
  {
   "method" : "VR.GetLanguage"
  }
 }
}
```


GetSupportedLanguages

Type Function
Sender SDL
Purpose Get the supported VR languages

REQUEST

PARAMETERS

NAME	TYPE	MANDATORY	ADDITIONAL

RESPONSE

PARAMETERS

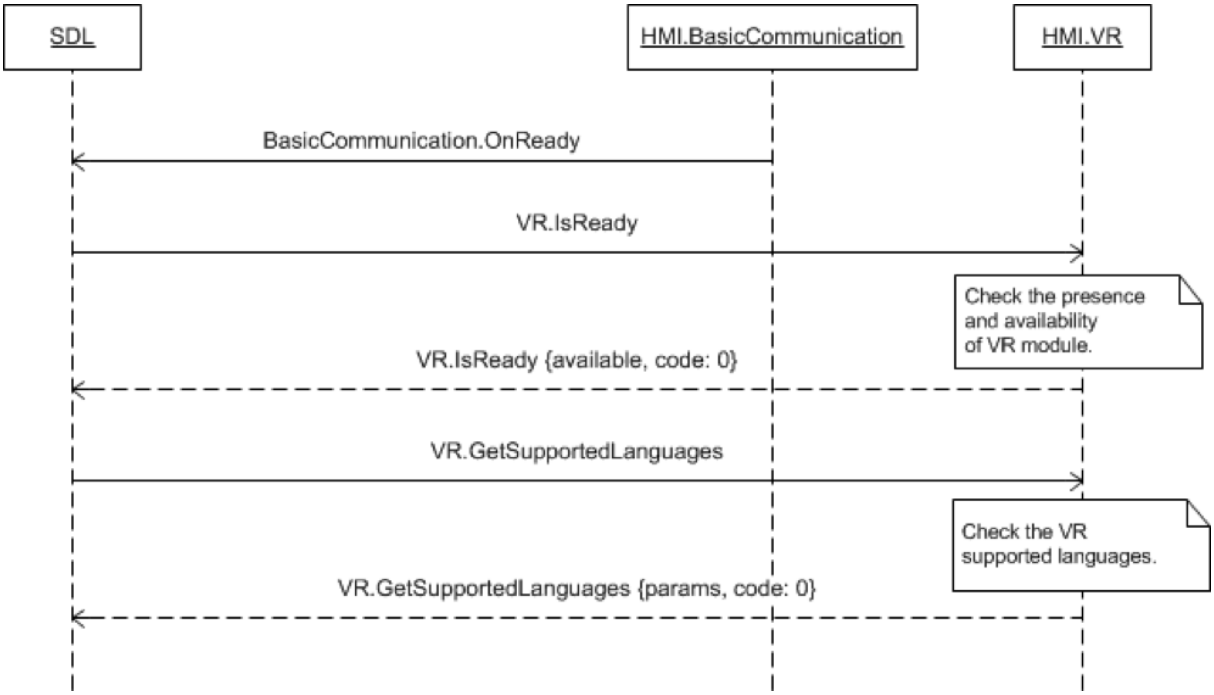
NAME	TYPE	MANDATORY	ADDITIONAL
languages	Common.Language	true	array: true minsize: 1 maxsize: 100

Sequence Diagrams

SEQUENCE DIAGRAM

GetSupportedLanguages

View Diagram



Example Request

```
{
 "id" : 18,
 "jsonrpc" : "2.0",
 "method" : "VR.GetSupportedLanguages"
}
```

Example Response

```
{
 "id" : 18,
 "jsonrpc" : "2.0",
 "result" :
 {
  "languages" : [AR-SA, DE-DE, EN-GB, EN-US, ES-ES, FR-FR, IT-IT],
  "code" : 0,
  "method" : "VR.GetSupportedLanguages"
 }
}
```

Example Error


```
{
 "id" : 18,
 "jsonrpc" : "2.0",
 "error" :
 {
  "code" : 11,
  "message" : "The data sent is invalid",
  "data" :
  {
   "method" : "VR.GetSupportedLanguages"
  }
 }
}
```


IsReady

Type Function
Sender SDL
Purpose Request ready state of VR Module

REQUEST

NOTE

1. SDL sends `VR.IsReady` request after HMI confirms its readiness via `BC.OnReady` notification.
2. If HMI responds with `"available":false`, SDL will not further communicate over VR interface with HMI.
3. If HMI does not respond during SDL's default timeout, SDL will continue to send RPCs over VR interface to HMI.

PARAMETERS

The request does not have specific parameters.

RESPONSE

MUST

1. Check whether VR module is available and ready.

2. Respond correspondingly to results of this check.

PARAMETERS

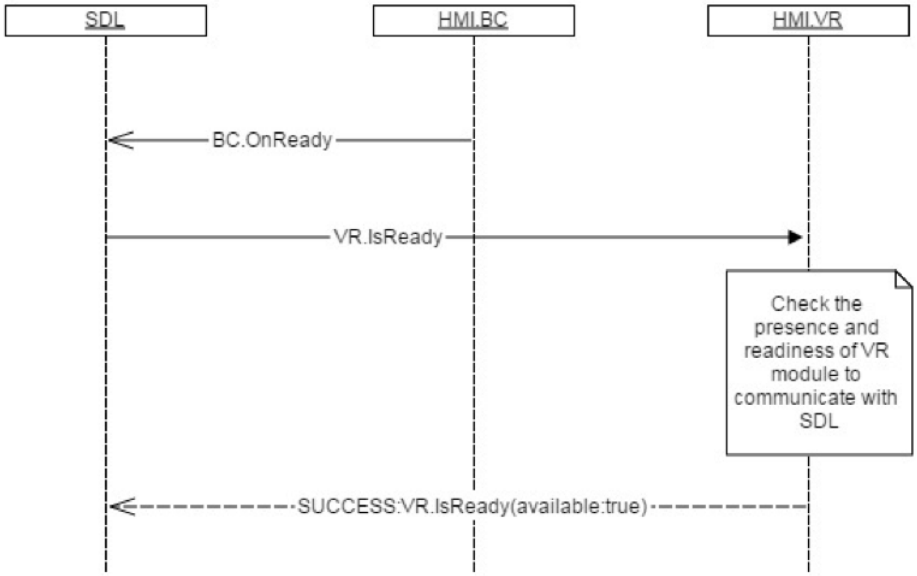
NAME	TYPE	MANDATORY	ADDITIONAL
available	Boolean	true	

Sequence Diagrams

SEQUENCE DIAGRAM

IsReady

View Diagram



Example Request

```
{
 "id" : 45,
 "jsonrpc" : "2.0",
 "method" : "VR.IsReady"
}
```

Example Response

```
{
 "id" : 45,
 "jsonrpc" : "2.0",
 "result" :
 {
  "availabe" : true,
  "code" : 0,
  "method" : "VR.IsReady"
 }
}
```

Example Error

```
{
 "id" : 45,
 "jsonrpc" : "2.0",
 "error" :
 {
  "code" : 11,
  "message" : "Invalid data",
  "data" :
  {
   "method" : "VR.IsReady"
  }
 }
}
```

OnCommand

Type Notification
Sender HMI
Purpose Inform SDL that an application's VR command was recognized.

Notification

PARAMETERS

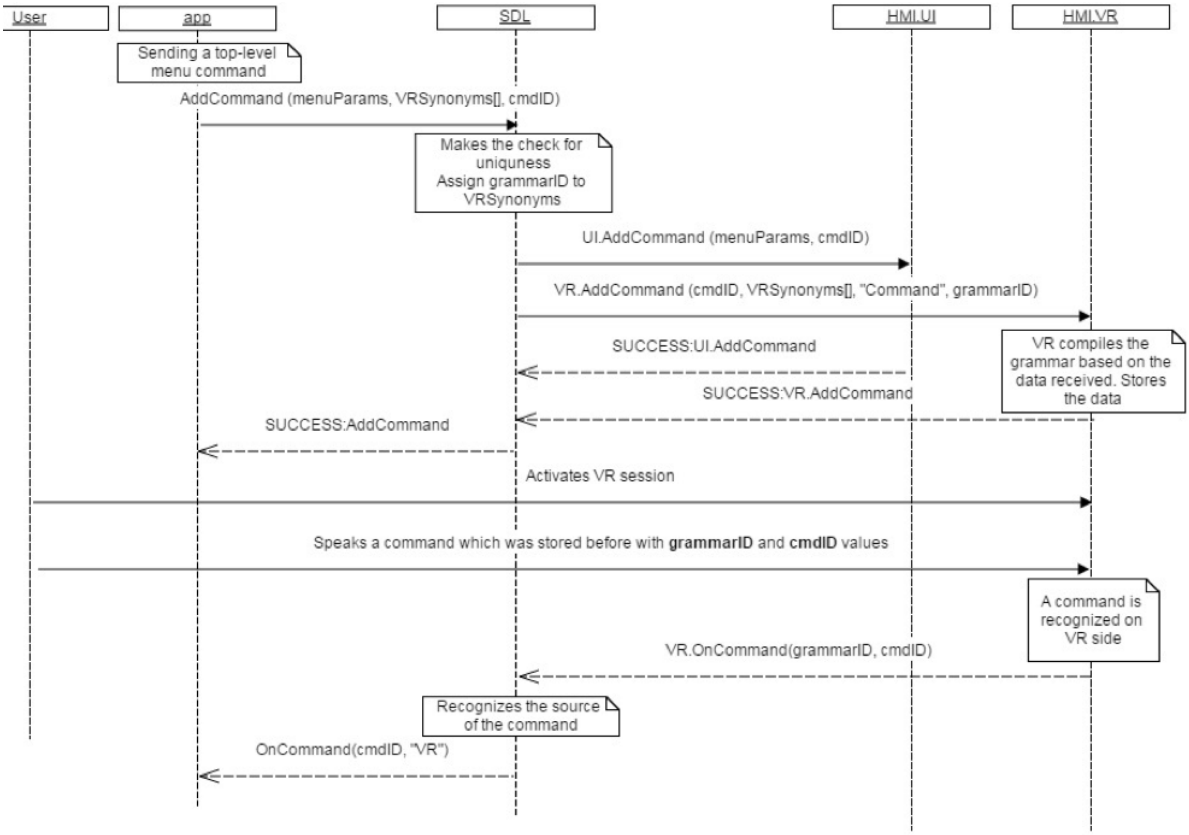
NAME	TYPE	MANDATORY	ADDITIONAL
cmdID	Integer	true	minvalue: 0 maxvalue: 2000000000
appID	Integer	true	

Sequence Diagrams

SEQUENCE DIAGRAM

OnCommand

View Diagram



JSON EXAMPLE NOTIFICATION

```
{
 "jsonrpc" : "2.0",
 "method" : "VR.OnCommand",
 "params" :
 {
  "cmdID" : 4365,
   "grammarID" : 11,
   "appID" : 12564
 }
}
```

OnLanguageChange

Type

Notification

Sender

HMI

Purpose

Inform SDL that the language of the VR engine has changed.

Notification

PARAMETERS

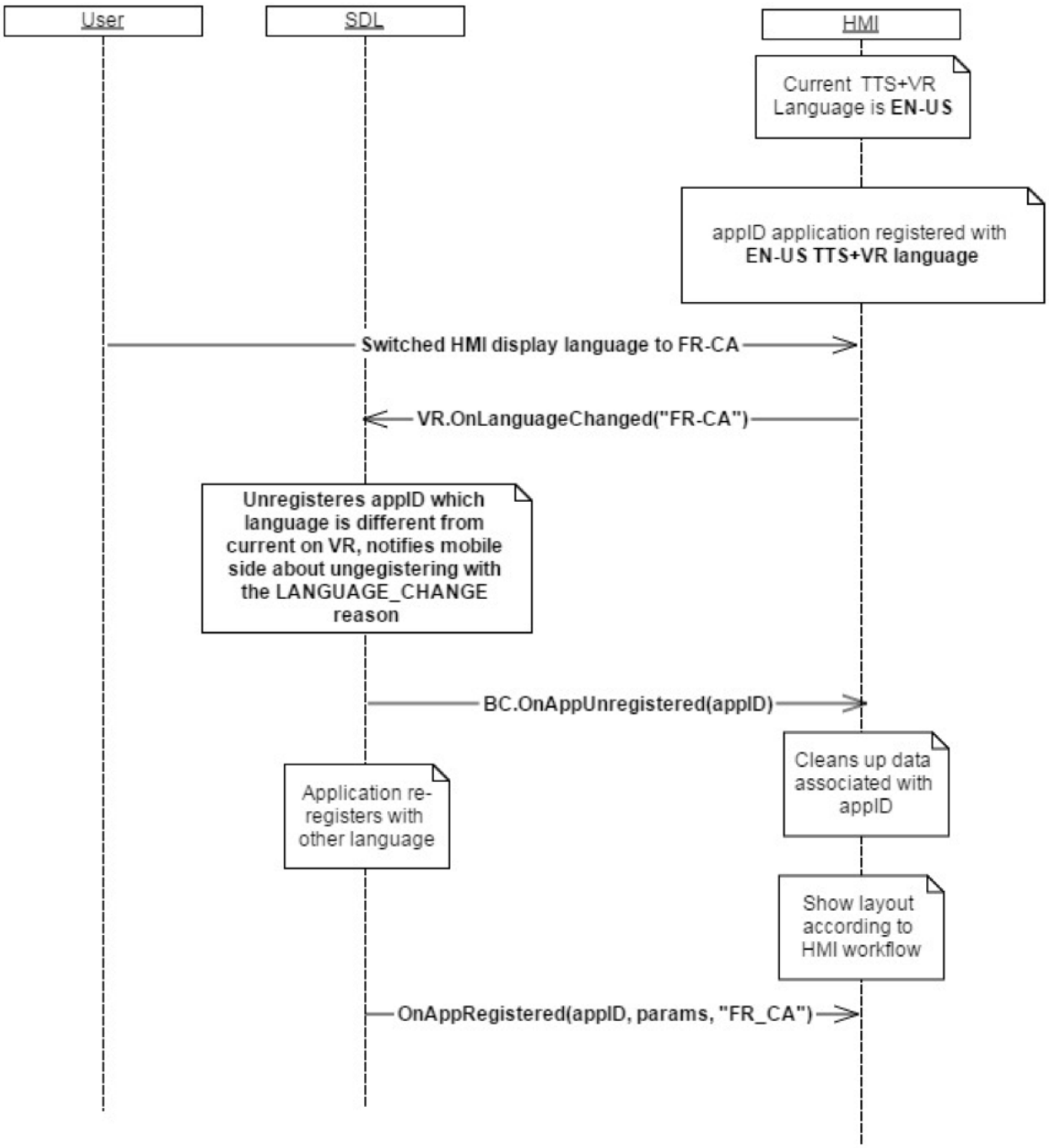
NAME	TYPE	MANDATORY	ADDITIONAL
language	Common.Language	true	

Sequence Diagrams

SEQUENCE DIAGRAM

OnLanguageChange

View Diagram



JSON EXAMPLE NOTIFICATION

```
{
 "jsonrpc" : "2.0",
 "method" : "VR.OnLanguageChange",
 "params" :
 {
  "language" : "IT-IT"
 }
}
```

PerformInteraction

Type Function
Sender SDL
Purpose Perform a VR interaction with the User.

REQUEST

PARAMETERS

NAME	TYPE	MANDATORY	ADDITIONAL
helpPrompt	Common.TTSChunk	false	array: true minsize: 1 maxsize: 100
initialPrompt	Common.TTSChunk	false	array: true minsize: 1 maxsize: 100
timeoutPrompt	Common.TTSChunk	false	array: true minsize: 1 maxsize: 100
timeout	Integer	true	
grammarID	Integer	false	array: true minsize: 1 maxsize: 100 minvalue: 0 maxvalue: 2000000000
appID	Integer	true	

RESPONSE

PARAMETERS

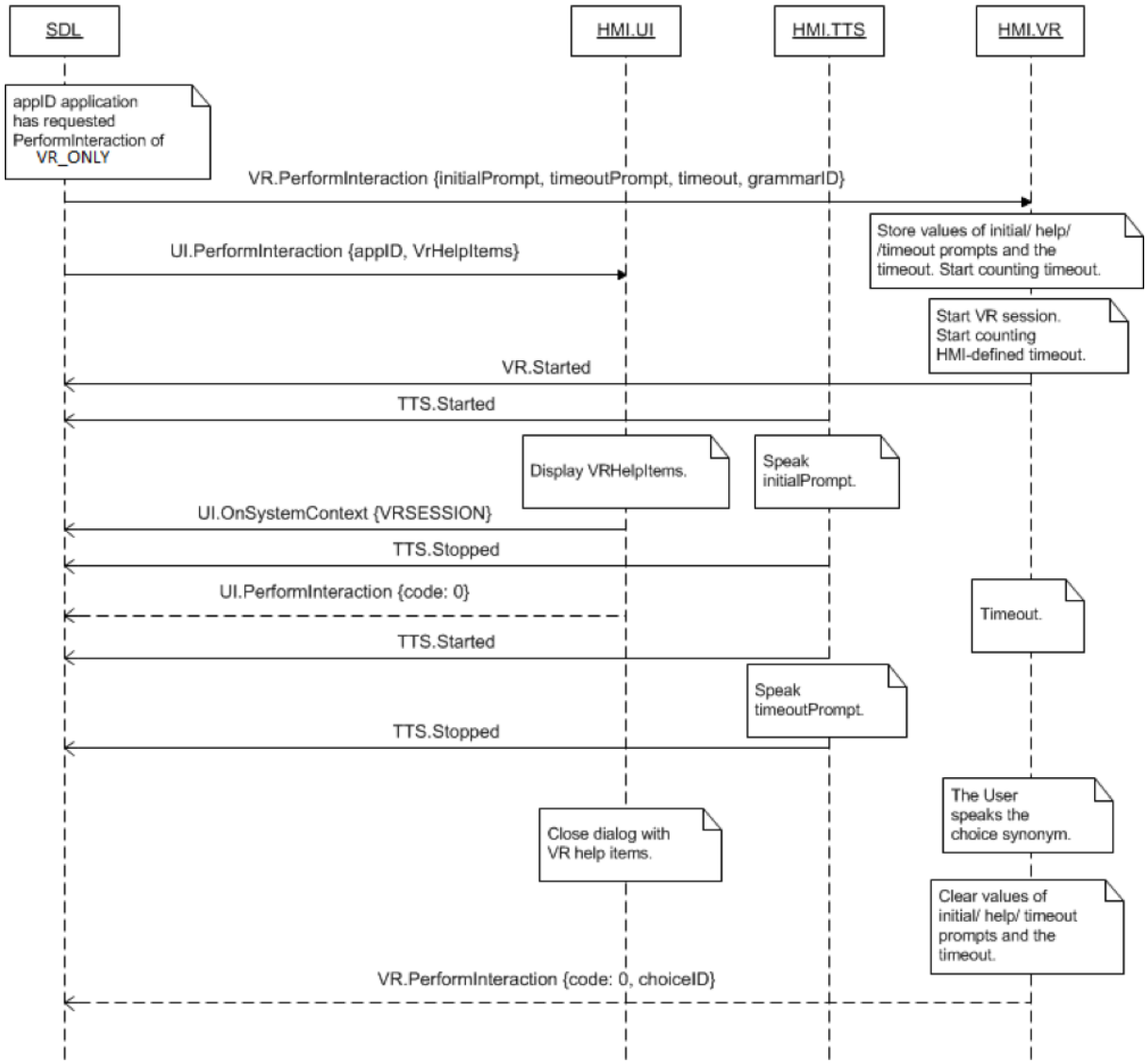
NAME	TYPE	MANDATORY	ADDITIONAL
choiceID	Integer	false	minvalue: 0 maxvalue: 2000000000

Sequence Diagrams

SEQUENCE DIAGRAM

PerformInteraction in VR Mode completed successfully

View Diagram

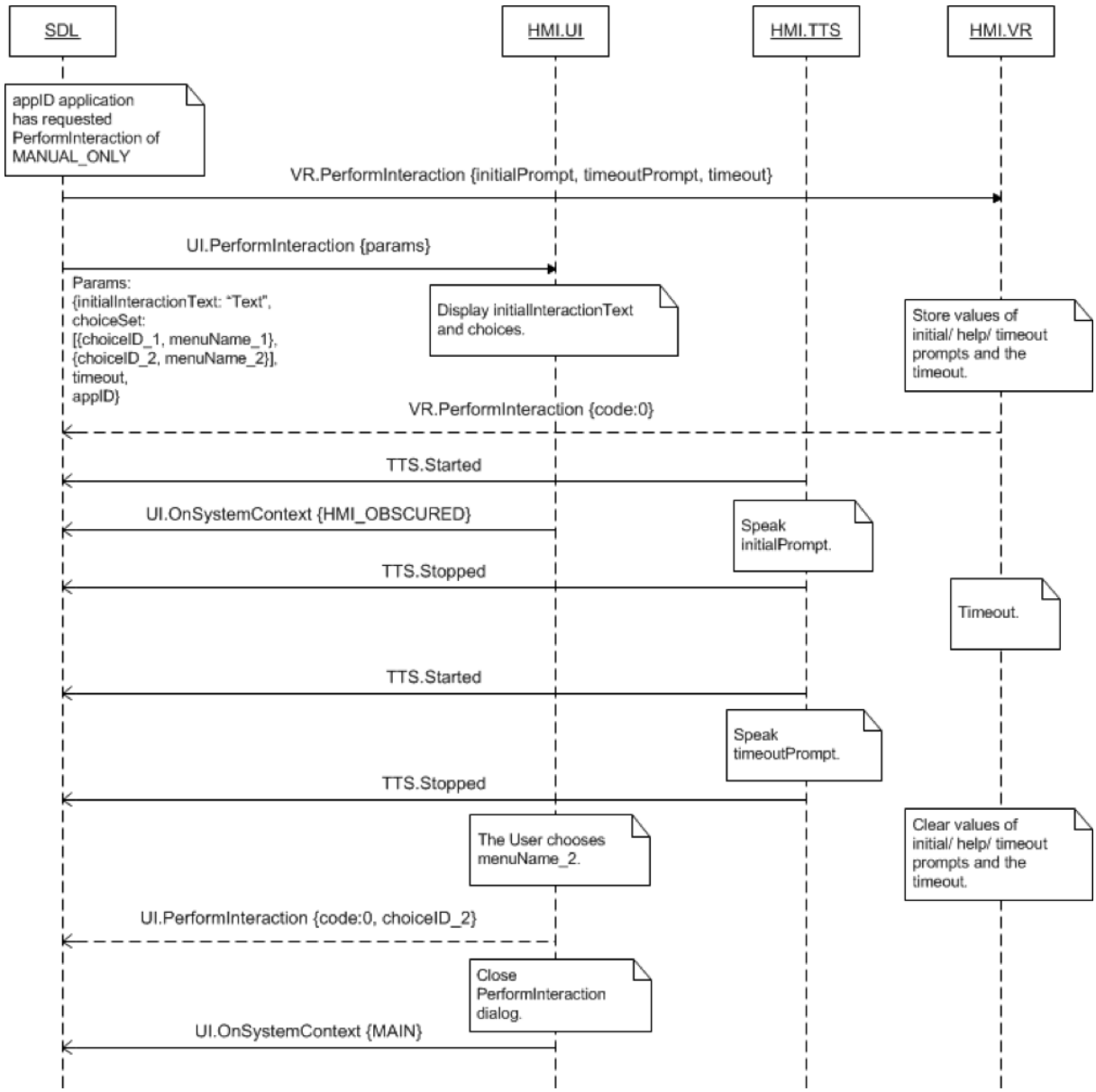


SEQUENCE DIAGRAM

PerformInteraction in Manual Only Mode completed successfully

---

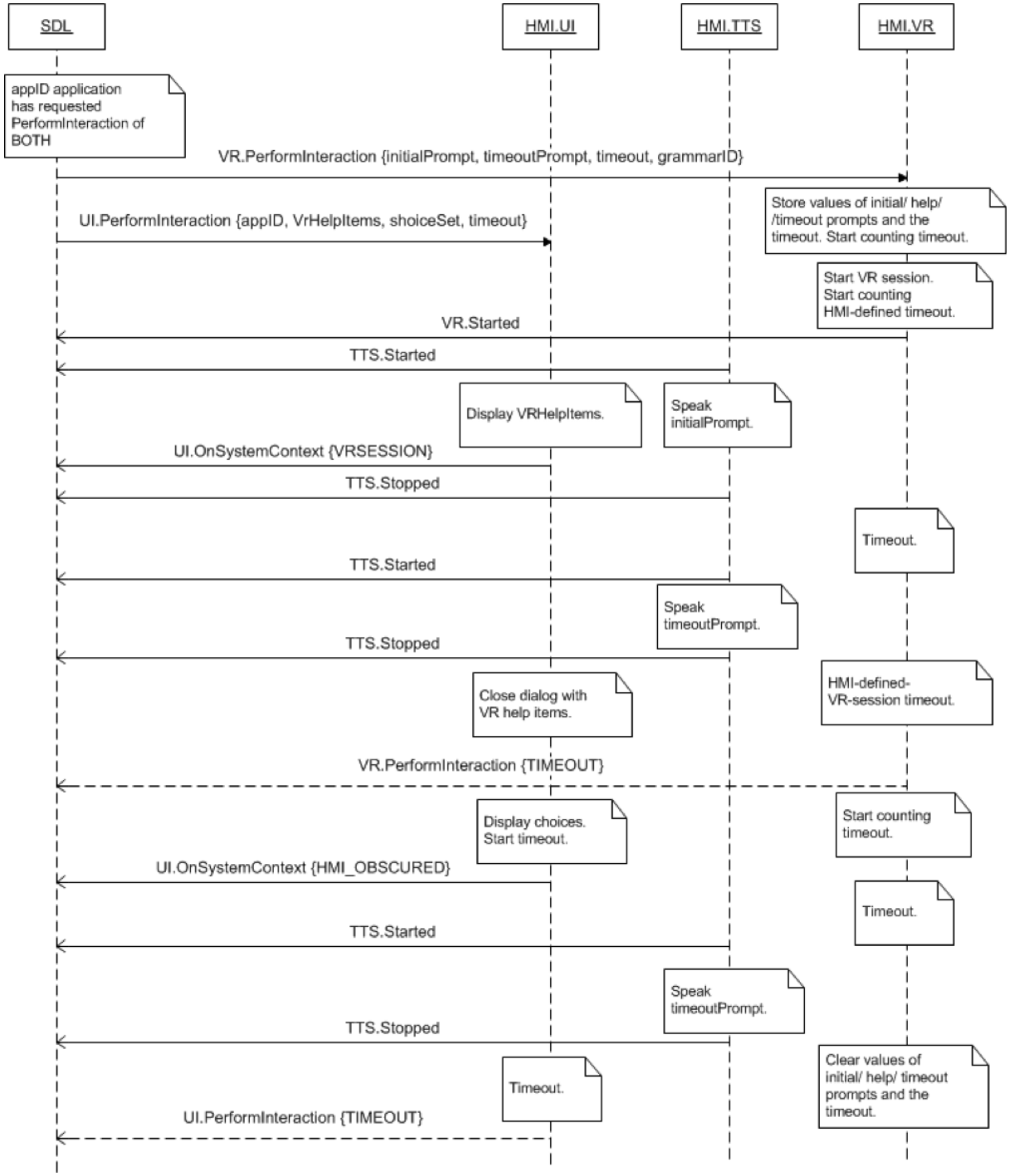
View Diagram



SEQUENCE DIAGRAM

PerformInteraction in Both Mode times out

View Diagram


SDL
HMI.UI
HMI.TTS
HMI.VR
appID application has requested PerformInteraction of BOTH
VR.PerformInteraction {initialPrompt, timeoutPrompt, timeout, grammarID}
Store values of initial/ help/ /timeout prompts and the timeout. Start counting timeout.
UI.PerformInteraction {appID, VrHelpItems, shoiceSet, timeout}
Start VR session. Start counting HMI-defined timeout.
VR.Started
TTS.Started
Display VRHelpItems.
Speak initialPrompt.
UI.OnSystemContext {VRSESSION}
TTS.Stopped
Timeout.
TTS.Started
Speak timeoutPrompt.
TTS.Stopped
Close dialog with VR help items.
HMI-defined-VR-session timeout.
VR.PerformInteraction {TIMEOUT}
Display choices. Start timeout.
Start counting timeout.
UI.OnSystemContext {HMI_OBSCURED}
Timeout.
TTS.Started
Speak timeoutPrompt.
TTS.Stopped
Timeout.
Clear values of initial/ help/ timeout prompts and the timeout.
UI.PerformInteraction {TIMEOUT}

Example Request


```
{
 "id" : 79,
 "jsonrpc" : "2.0",
 "method" : "VR.PerformInteraction",
 "params" :
 {
  "initialPrompt" :
  [
   {
    "text" : "Please make your choice by voice",
    "type" : "TEXT"
   }
  ],

  "helpPrompt" :
  [
   {
    "text" : "Yes",
    "type" : "TEXT"
   },
   {
    "text" : "No",
    "type" : "TEXT"
   },
   {
    "text" : "Skip",
    "type" : "TEXT"
   }
  ],

  "timeoutPrompt" :
  [
   {
    "text" : "The time is about to run out",
    "type" : "TEXT"
   }
  ],

  "timeout" : 10000,
  "grammarID" : 245,
  "appID" : 101
 }
}
```

Example Response


```
{
 "id" : 79,
 "jsonrpc" : "2.0",
 "result" :
 {
  "choiceID" : 2416,
  "code" : 0,
  "method" : "VR.PerformInteraction"
 }
}
```


Example Error


```
{
 "id" : 79,
 "jsonrpc" : "2.0",
 "error" :
 {
  "code" : 10,
  "message" : "Interaction reached the maximum timeout and will be
closed",
  "data" :
  {
   "method" : "VR.PerformInteraction"
  }
 }
}
```

Started

Type Notification
Sender HMI
Purpose Inform SDL about the start of a VR session

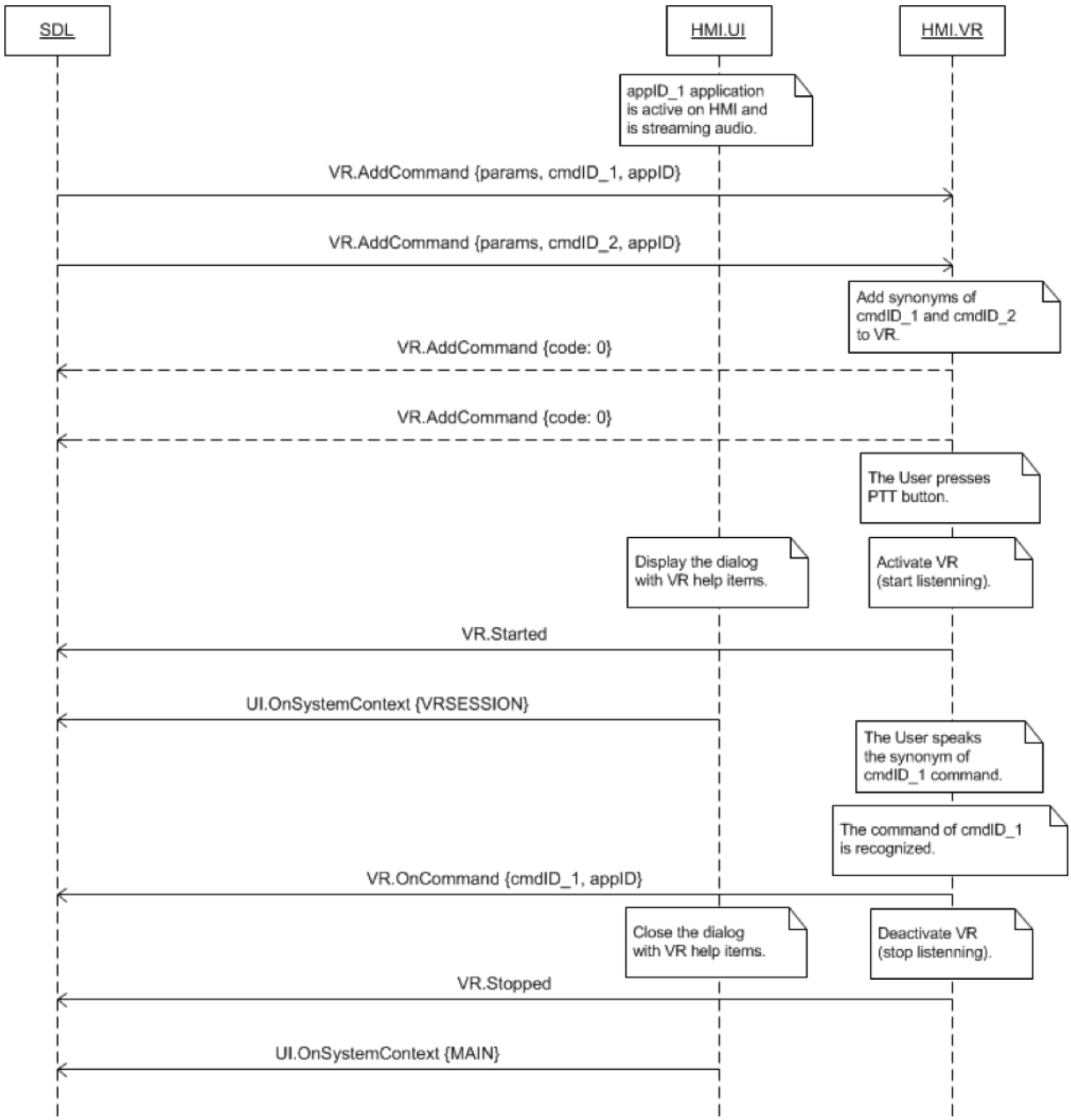
Notification

Sequence Diagrams

SEQUENCE DIAGRAM

Started on PTT Button Press

View Diagram



Example Notification

```json
{
"jsonrpc" : "2.0",
"method" : "VR.Started",
}
```

Stopped

Type Notification
Sender HMI
Purpose Inform SDL about the end of a VR session.

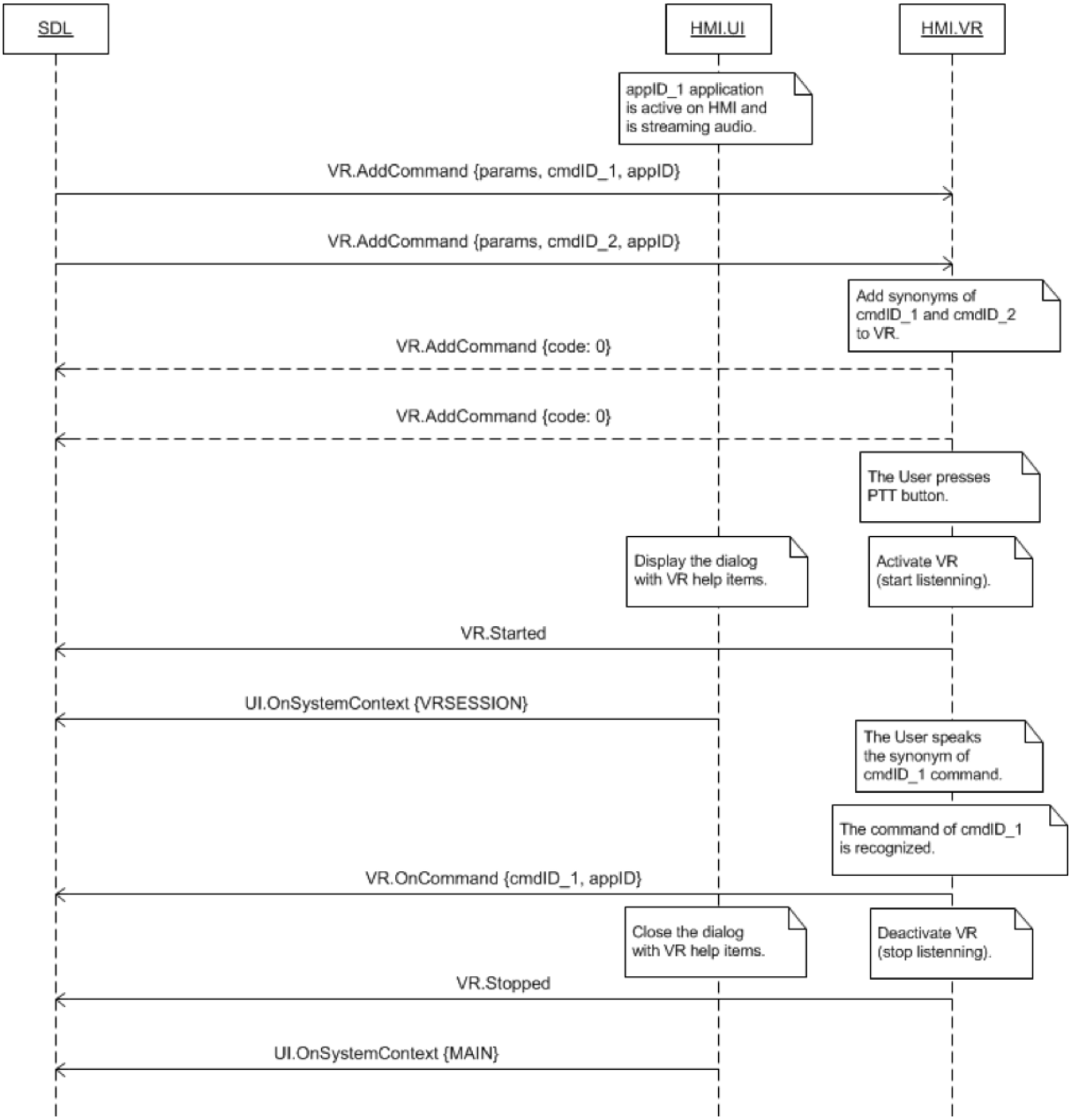
Notification

Sequence Diagrams

SEQUENCE DIAGRAM

Stopped when VR session ends

View Diagram



Example Notification

```
{
  "jsonrpc" : "2.0",
  "method" : "VR.Stopped",
}
```

GetCapabilities

Type Function
Sender SDL
Purpose Inform SDL of the Remote Control capabilities of HMI, invoked at system startup.

GetCapabilities is a request originated by SDL Core after receiving an `IsReady` Response from the Remote Control interface. If the HMI responds to GetCapabilities with the `RemoteControlCapabilities` parameter, this struct will overwrite any Remote Control related capabilities that are stored in the `hmi_capabilities.json` configuration file.

The `RemoteControlCapabilities` struct will be stored by SDL Core, and used when a mobile application performs a `GetSystemCapability` request.

REQUEST

PARAMETERS

This RPC has no additional parameter requirements

RESPONSE

PARAMETERS

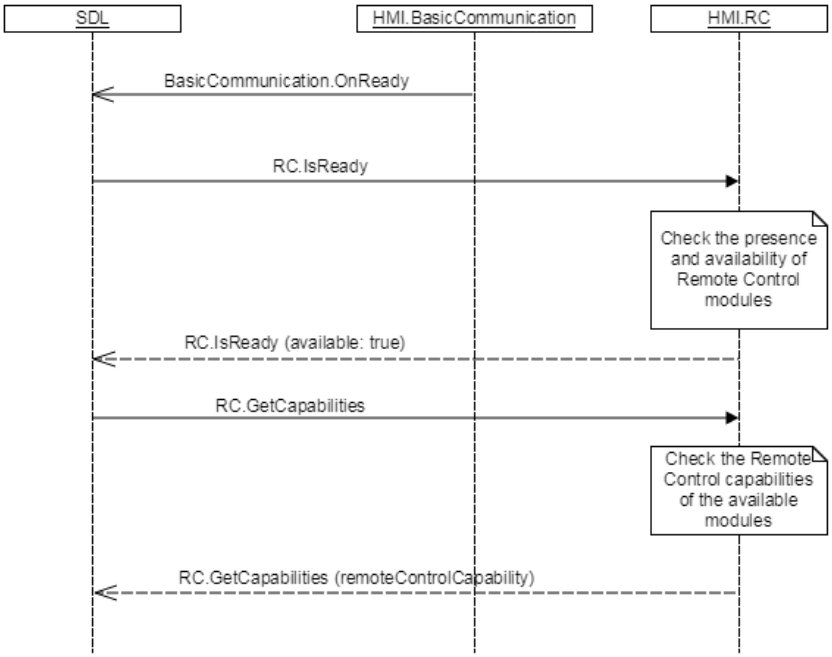
NAME	TYPE	MANDATORY	ADDITIONAL
remoteControlCapability	Common.RemoteControlCapabilities	false	See RemoteControlCapabilities, all available RC modules and buttons shall be returned

Sequence Diagrams

SEQUENCE DIAGRAM

GetCapabilities

View Diagram



Example Request

```
{
 "id":22,
 "jsonrpc":"2.0",
 "method":"RC.GetCapabilities"
}
```

Example Response

```
{
  "id": 22,
  "jsonrpc": "2.0",
  "result": {
    "code": 0,
    "method": "RC.GetCapabilities",
    "remoteControlCapability": {
      "buttonCapabilities": [
        {
          "longPressAvailable": true,
          "name": "AC_MAX",
          "shortPressAvailable": true,
          "upDownAvailable": true
        },
        {
          "longPressAvailable": true,
          "name": "AC",
          "shortPressAvailable": true,
          "upDownAvailable": true
        },
        {
          "longPressAvailable": true,
          "name": "RECIRCULATE",
          "shortPressAvailable": true,
          "upDownAvailable": true
        },
        {
          "longPressAvailable": true,
          "name": "FAN_UP",
          "shortPressAvailable": true,
          "upDownAvailable": true
        },
        {
          "longPressAvailable": true,
          "name": "FAN_DOWN",
          "shortPressAvailable": true,
          "upDownAvailable": true
        },
        {
          "longPressAvailable": true,
          "name": "TEMP_UP",
          "shortPressAvailable": true,
          "upDownAvailable": true
        },
        {
          "longPressAvailable": true,
          "name": "TEMP_DOWN",
          "shortPressAvailable": true,
          "upDownAvailable": true
```

```
        },
        {
          "longPressAvailable": true,
          "name": "DEFROST_MAX",
          "shortPressAvailable": true,
          "upDownAvailable": true
        },
        {
          "longPressAvailable": true,
          "name": "DEFROST",
          "shortPressAvailable": true,
          "upDownAvailable": true
        },
        {
          "longPressAvailable": true,
          "name": "DEFROST_REAR",
          "shortPressAvailable": true,
          "upDownAvailable": true
        },
        {
          "longPressAvailable": true,
          "name": "UPPER_VENT",
          "shortPressAvailable": true,
          "upDownAvailable": true
        },
        {
          "longPressAvailable": true,
          "name": "LOWER_VENT",
          "shortPressAvailable": true,
          "upDownAvailable": true
        },
        {
          "longPressAvailable": true,
          "name": "VOLUME_UP",
          "shortPressAvailable": true,
          "upDownAvailable": true
        },
        {
          "longPressAvailable": true,
          "name": "VOLUME_DOWN",
          "shortPressAvailable": true,
          "upDownAvailable": true
        },
        {
          "longPressAvailable": true,
          "name": "EJECT",
          "shortPressAvailable": true,
          "upDownAvailable": true
        },
        {
          "longPressAvailable": true,
          "name": "SOURCE",
          "shortPressAvailable": true,
          "upDownAvailable": true
```

```
      },
      {
        "longPressAvailable": true,
        "name": "SHUFFLE",
        "shortPressAvailable": true,
        "upDownAvailable": true
      },
      {
        "longPressAvailable": true,
        "name": "REPEAT",
        "shortPressAvailable": true,
        "upDownAvailable": true
      }
    ],
    "climateControlCapabilities": [
      {
        "acEnableAvailable": true,
        "acMaxEnableAvailable": true,
        "autoModeEnableAvailable": true,
        "circulateAirEnableAvailable": true,
        "currentTemperatureAvailable": true,
        "defrostZone": [
          "FRONT",
          "REAR",
          "ALL",
          "NONE"
        ],
        "defrostZoneAvailable": true,
        "desiredTemperatureAvailable": true,
        "dualModeEnableAvailable": true,
        "fanSpeedAvailable": true,
        "moduleName": "Climate",
        "ventilationMode": [
          "UPPER",
          "LOWER",
          "BOTH",
          "NONE"
        ],
        "ventilationModeAvailable": true
      }
    ],
    "radioControlCapabilities": [
      {
        "availableHDsAvailable": true,
        "hdChannelAvailable": true,
        "moduleName": "Radio",
        "radioBandAvailable": true,
        "radioEnableAvailable": true,
        "radioFrequencyAvailable": true,
        "rdsDataAvailable": true,
        "signalChangeThresholdAvailable": true,
        "signalStrengthAvailable": true,
        "stateAvailable": true
      }
```

```
      ]
    }
  }
}
```

Example Error

```
{
 "id" : 22,
 "jsonrpc" : "2.0",
 "error" :
 {
  "code" : 22,
  "message" : "During API call the unknown error has occurred",
  "data" :
  {
   "method" : "RC.GetCapabilities"
  }
 }
}
```

GetInteriorVehicleData

Type Function
Sender SDL

Purpose

To read RC module status data. The same function is used to subscribe/ unsubscribe to RC module status/setting change notifications.

GetInteriorVehicleData is a request originated by a Remote Control Mobile Application.
If the parameter `subscribe` is set to `true`, the mobile application has requested to subscribe to the module data defined by the `moduleType` parameter.
SDL maintains the `moduleType` subscription status as a whole. SDL needs to subscribe to a module if there is at least one app that subscribes to the module. SDL needs to unsubscribe from a module if no apps subscribe to the module.

SDL forwards a GetInteriorVehicleData request to HMI only if there is no cached data available for the requested `moduleType` or it needs to unsubscribe to the module from HMI.
Otherwise, SDL responds to the request with the cached data without forwarding it to HMI.

The HMI should only return interior vehicle data that corresponds to the request `moduleType`. For example, if `moduleType = CLIMATE`, only return `ClimateControlData` and do not return `RadioControlData`.

REQUEST

GetInteriorVehicleData is a request originated by a Remote Control Mobile Application. The HMI should only return interior vehicle data that corresponds to the request module type.
For example, if `moduleType = CLIMATE`, only return `ClimateControlData` and do not return `RadioControlData`.

If the parameter `subscribe` is set to true, the mobile application has requested to subscribe to the module data defined by the `moduleType` parameter.

PARAMETERS

NAME	TYPE	MANDATORY	ADDITIONAL
moduleType	Common.ModuleType	true	
subscribe	Boolean	false	defvalue="false"

RESPONSE

HMI must return in GetInteriorVehicleData_response the current value of the display mode used in HMI if `moduleType = HMI_SETTINGS` .

PARAMETERS

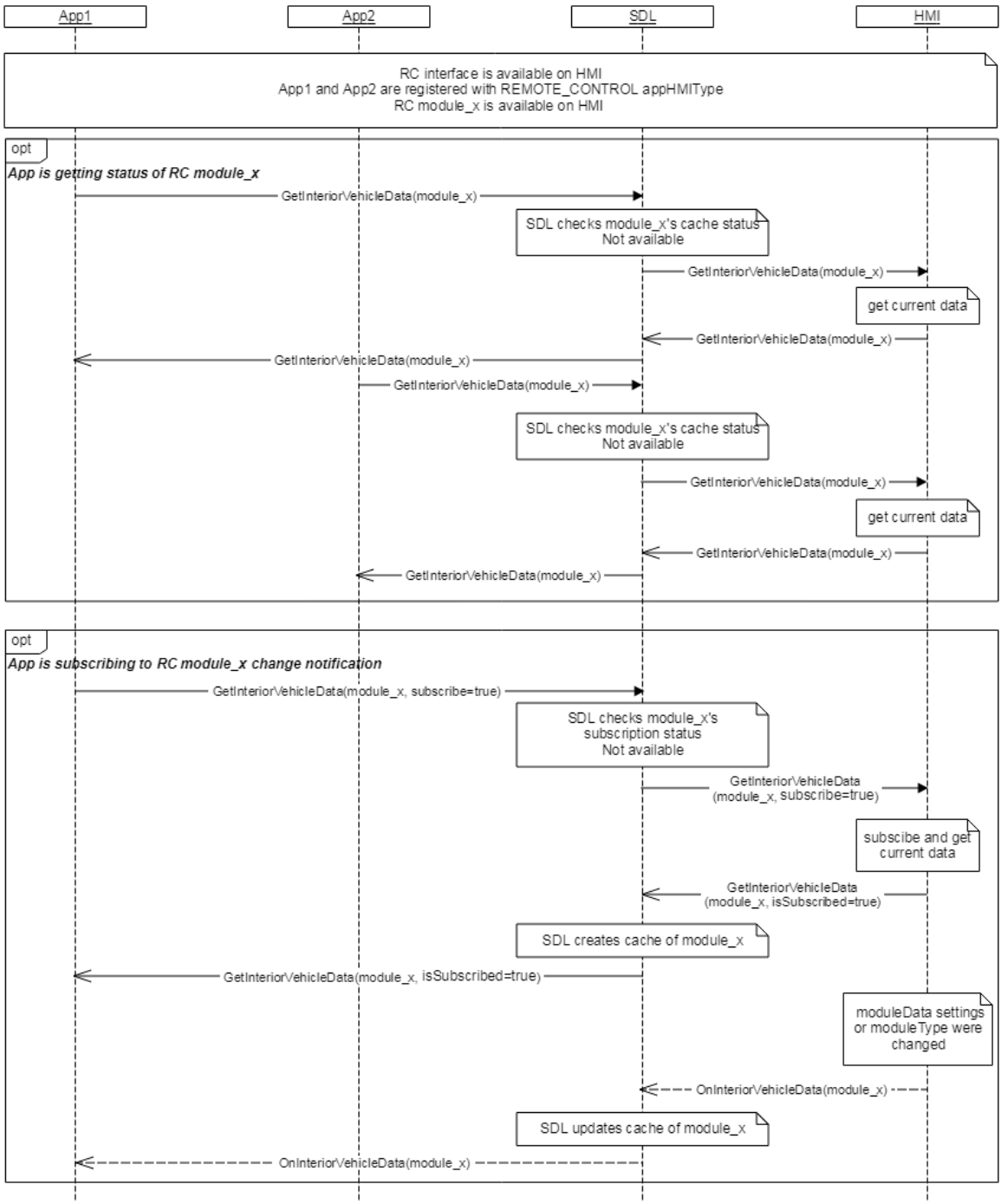
NAME	TYPE	MANDATORY	ADDITIONAL
moduleData	Common.ModuleData	true	
isSubscribed	Boolean	false	

Sequence Diagrams

SEQUENCE DIAGRAM

GetInteriorVehicleData

View Diagram



Example Request

```
{
    "id": 122,
    "jsonrpc": "2.0",
    "method": "RC.GetInteriorVehicleData",
    "params": {
        "moduleType": "CLIMATE",
        "subscribe": true
    }
}
```

Example Response


```
{
   "id": 122,
   "jsonrpc": "2.0",
   "result": {
      "code": 0,
      "isSubscribed": true,
      "method": "RC.GetInteriorVehicleData",
      "moduleData": {
         "climateControlData": {
            "acEnable": true,
            "acMaxEnable": true,
            "autoModeEnable": true,
            "circulateAirEnable": true,
            "currentTemperature": {
               "unit": "FAHRENHEIT",
               "value": 20.1
            },
            "defrostZone": "FRONT",
            "desiredTemperature": {
               "unit": "CELSIUS",
               "value": 10.5
            },
            "dualModeEnable": true,
            "fanSpeed": 50,
            "ventilationMode": "BOTH"
         },
         "moduleType": "CLIMATE"
      }
   }
}
```

Example Error


```
{
   "error": {
      "code": 2,
      "data": {
         "method": "RC.GetInteriorVehicleData"
      },
      "message": "Unknown module type"
   },
   "id": 122,
   "jsonrpc": "2.0"
}
```


GetInteriorVehicleDataConsent

Type Function
Sender SDL
Purpose Request for driver's permission to RC module for the specified application

RC.GetInteriorVehicleDataConsent must be sent to HMI in case application in HMILevel FULL requested access to RC module that is already in use by another application.
HMI is expected to display a permission prompt to the driver showing the RC module and app details (for example, app's name).
The driver is expected to have an ability to grant or deny the permission.

MUST

1. HMI must prompt user to make selection of resource allocation
2. Respond to SDL with user choice within RC.GetInteriorVehicleDataConsent response
3. If user didn't make choice send GENERIC_ERROR to SDL

REQUEST

PARAMETERS

NAME	TYPE	MANDATORY	ADDITIONAL	DESCRIPTION
moduleType	Common.ModuleType	true		The module type that the app requests to control
appID	Integer	true		ID of the application that triggers the permission prompt

RESPONSE

PARAMETERS

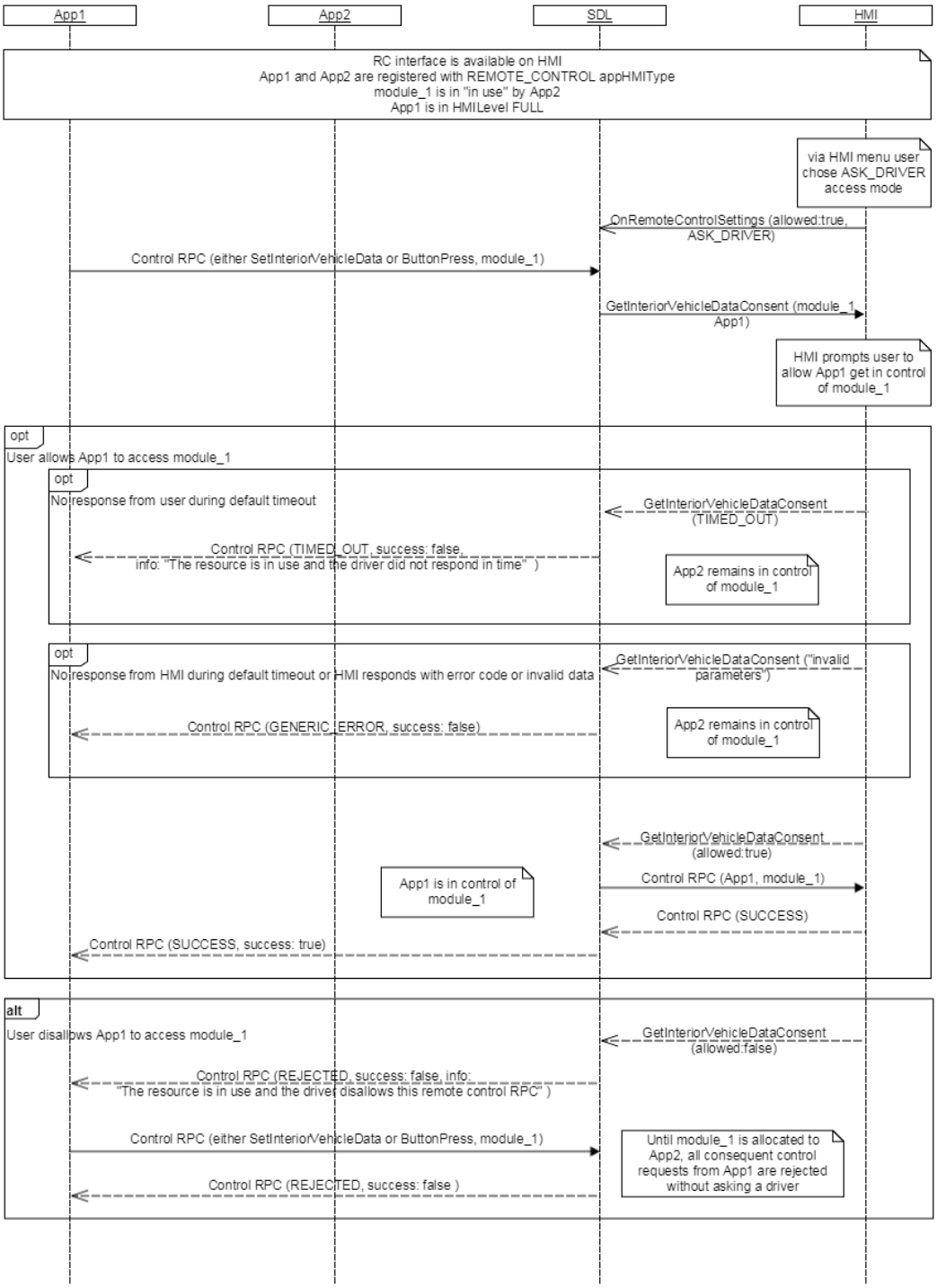
NAME	TYPE	MANDATORY	ADDITIONAL	DESCRIPTION
allowed	Boolean	true		"true" - if the driver grants the permission for controlling to the named app; "false" - in case the driver denies the permission for controlling to the named app.

Sequence Diagrams

SEQUENCE DIAGRAM

GetInteriorVehicleDataConsent

View Diagram



Example Request

```
{
    "id": 38,
    "jsonrpc": "2.0",
    "method": "RC.GetInteriorVehicleDataConsent",
    "params": {
        "appID": 2032286926,
        "moduleType": "CLIMATE"
    }
}
```

Example Response

```
{
    "id": 38,
    "jsonrpc": "2.0",
    "result": {
        "allowed": true,
        "code": 0,
        "method": "RC.GetInteriorVehicleDataConsent"
    }
}
```

Example Error


```
{
   "error": {
      "code": 10,
      "data": {
         "method": "RC.GetInteriorVehicleDataConsent"
      },
      "message": "No response from driver"
   },
   "id": 38,
   "jsonrpc": "2.0"
}
```


IsReady

Type Function
Sender SDL
Purpose Provide information to SDL about presence of any of remote controllable module and its readiness to cooperate with SDL, invoked at SDL system start up.

REQUEST

PARAMETERS

The request does not have specific parameters.

NOTE

1. SDL sends `RC.IsReady` request after HMI confirms its readiness via `BC.OnReady` notification.
2. If HMI responds with `"available":false`, SDL will not further communicate over RC interface with HMI.
3. If HMI does not respond during SDL's default timeout, SDL will continue to send RPCs over RC interface to HMI.

RESPONSE

MUST

1. Check whether RC component is available and ready.

2. Respond correspondingly to results of this check.

PARAMETERS

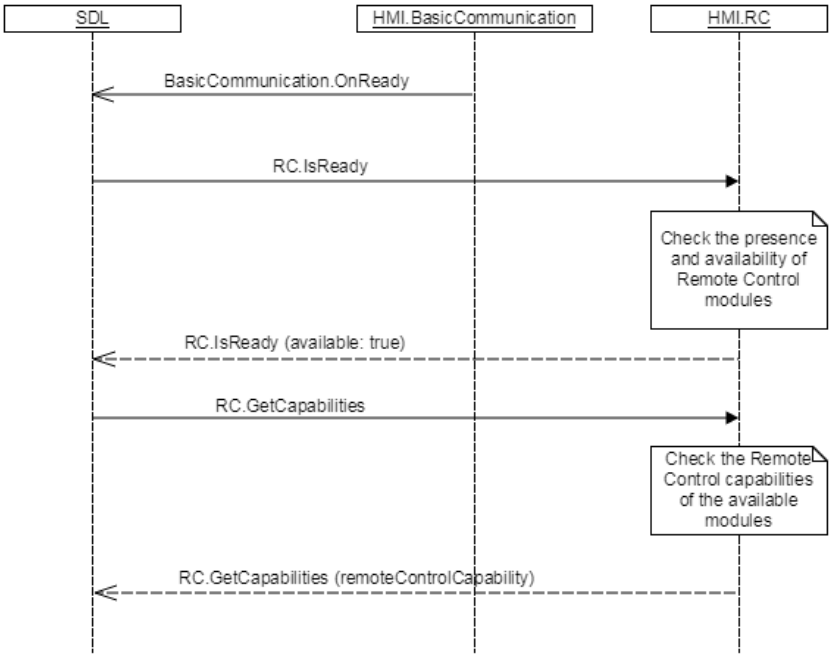
NAME	TYPE	MANDATORY	DESCRIPTION
available	Boolean	true	Must be true if vehicle RC modules are present and ready to communicate with SDL.

Sequence Diagrams

SEQUENCE DIAGRAM

IsReady

View Diagram



Example Request

```
{
 "id" : 8,
 "jsonrpc" : "2.0",
 "method" : "RC.IsReady"
}
```

Example Response

```
{
 "id" : 8,
 "jsonrpc" : "2.0",
 "result" :
 {
  "available" : true,
  "code" : 0,
  "method" : "RC.IsReady"
 }
}
```

Example Error

```
{
 "id" : 8,
 "jsonrpc" : "2.0",
 "error" :
 {
  "code" : 22,
  "message" : " HMI doesn’t have the information about RC availability
or some failure occurred ",
  "data" :
  {
   "method" : "RC.IsReady"
  }
 }
}
```

OnInteriorVehicleData

Type

Notification

Sender

HMI

Purpose

Inform SDL about any changes in RC module settings.

MUST

The HMI must send RC.OnInteriorVehicleData notification to SDL when module settings were changed in case of button press event or after SetInteriorVehicledata request was processed.

The HMI must send RC.OnInteriorVehicleData notification with the current display mode to all mobile RC applications that are subscribed to the HMI settings

if the driver (or other applications) or HMI itself change the display mode in the HMI settings.

Notification

PARAMETERS

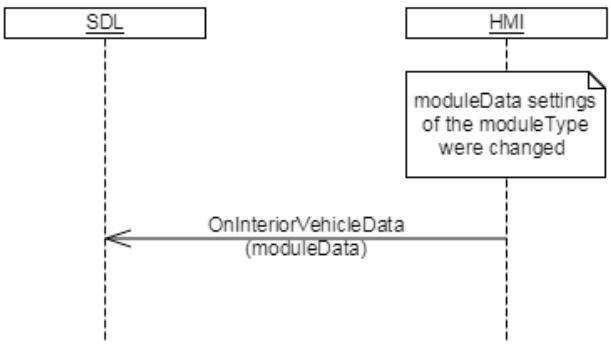
NAME	TYPE	MANDATORY	ADDITIONAL
moduleData	Common.ModuleData	true	

Sequence Diagrams

SEQUENCE DIAGRAM

OnInteriorVehicledata

View Diagram



JSON EXAMPLE NOTIFICATION


```
{
   "jsonrpc": "2.0",
   "method": "RC.OnInteriorVehicleData",
   "params": {
      "moduleData": {
         "climateControlData": {
            "acEnable": false,
            "acMaxEnable": false,
            "autoModeEnable": true,
            "circulateAirEnable": false,
            "currentTemperature": {
               "unit": "FAHRENHEIT",
               "value": 44.3
            },
            "defrostZone": "ALL",
            "desiredTemperature": {
               "unit": "CELSIUS",
               "value": 22.6
            },
            "dualModeEnable": true,
            "fanSpeed": 65,
            "ventilationMode": "UPPER"
         },
         "moduleType": "CLIMATE"
      }
   }
}
```


OnRemoteControlSettings

Type

Notification

Sender

HMI

Purpose

Notification about remote-control settings and/or access mode changed. Sent after User's choice through HMI.

RC.OnRemoteControlSettings is expected to be sent by HMI on system start up and every time user is changig remote control settings.
In case HMI does not send the notification on system start up, the default settings must be applied such as: remote control is allowed and access mode is "AUTO_ALLOW"

When user chose through HMI to disable remote control functionality the notification with allowed:false parameter must be sent to SDL

Notification

PARAMETERS

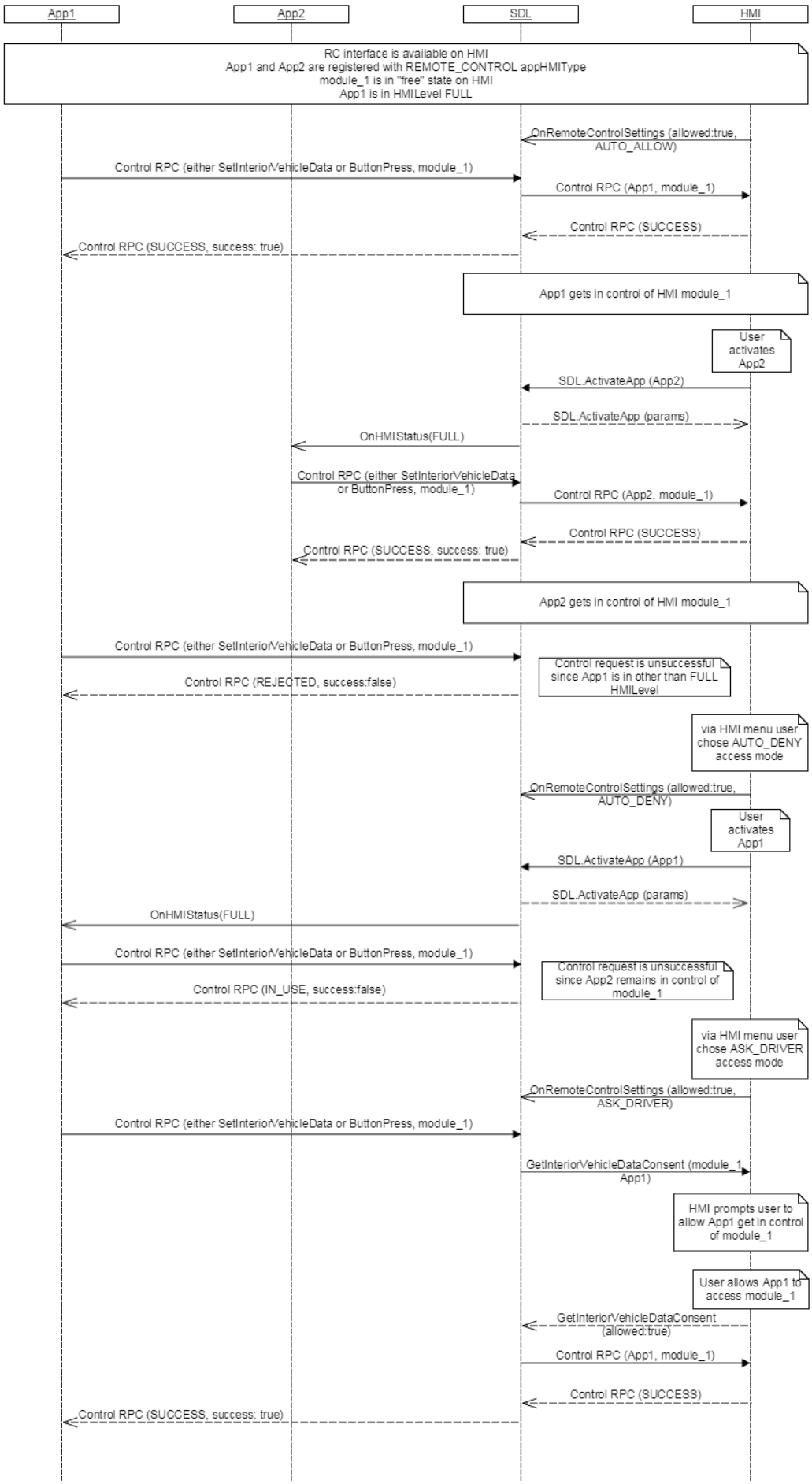
NAME	TYPE	MANDATORY	ADDITIONAL	DESCRIPTION
allowed	Boolean	false		If "true" - RC is allowed; if "false" - RC is disallowed
accessMode	Common.RCAccessMode	false		The remote control access mode specified by the driver via HMI

Sequence Diagrams

SEQUENCE DIAGRAM

OnRemoteControlSettings

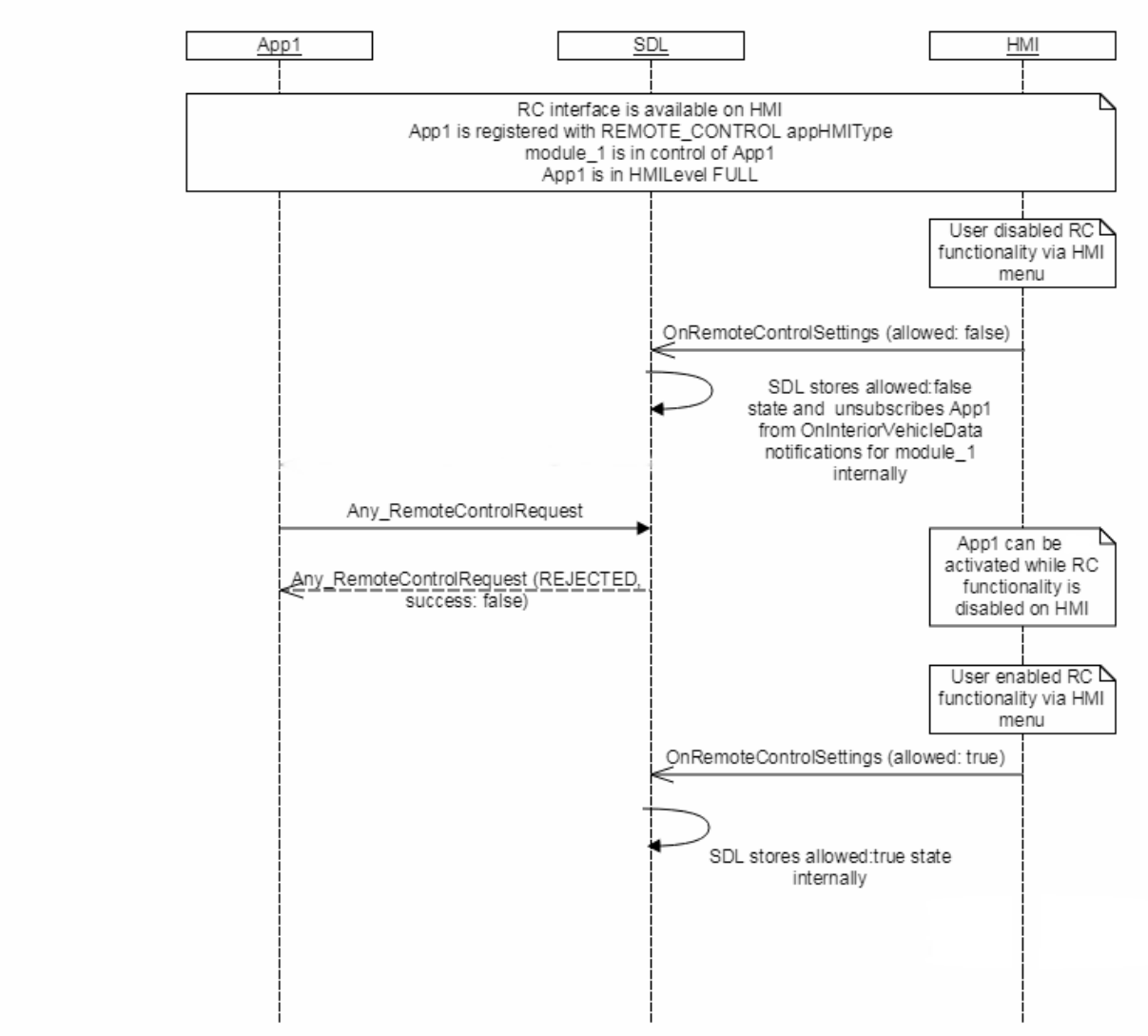
View Diagram


App1
App2
SDL
HMI
RC interface is available on HMI
App1 and App2 are registered with REMOTE_CONTROL appHMIType
module_1 is in "free" state on HMI
App1 is in HMILevel FULL
OnRemoteControlSettings (allowed:true, AUTO_ALLOW)
Control RPC (either SetInteriorVehicleData or ButtonPress, module_1)
Control RPC (App1, module_1)
Control RPC (SUCCESS)
Control RPC (SUCCESS, success: true)
App1 gets in control of HMI module_1
User activates App2
SDL.ActivateApp (App2)
SDL.ActivateApp (params)
OnHMIStatus(FULL)
Control RPC (either SetInteriorVehicleData or ButtonPress, module_1)
Control RPC (App2, module_1)
Control RPC (SUCCESS)
Control RPC (SUCCESS, success: true)
App2 gets in control of HMI module_1
Control RPC (either SetInteriorVehicleData or ButtonPress, module_1)
Control request is unsuccessful since App1 is in other than FULL HMILevel
Control RPC (REJECTED, success:false)
via HMI menu user chose AUTO_DENY access mode
OnRemoteControlSettings (allowed:true, AUTO_DENY)
User activates App1
SDL.ActivateApp (App1)
SDL.ActivateApp (params)
OnHMIStatus(FULL)
Control RPC (either SetInteriorVehicleData or ButtonPress, module_1)
Control request is unsuccessful since App2 remains in control of module_1
Control RPC (IN_USE, success:false)
via HMI menu user chose ASK_DRIVER access mode
OnRemoteControlSettings (allowed:true, ASK_DRIVER)
Control RPC (either SetInteriorVehicleData or ButtonPress, module_1)
GetInteriorVehicleDataConsent (module_1, App1)
HMI prompts user to allow App1 get in control of module_1
User allows App1 to access module_1
GetInteriorVehicleDataConsent (allowed:true)
Control RPC (App1, module_1)
Control RPC (SUCCESS)
Control RPC (SUCCESS, success: true)

SEQUENCE DIAGRAM

OnRemoteControlSettings disabling RC

View Diagram



JSON EXAMPLE NOTIFICATION

```
{
    "jsonrpc": "2.0",
    "method": "RC.OnRemoteControlSettings",
    "params": {
        "accessMode": "ASK_DRIVER",
        "allowed": true
    }
}
```

SetInteriorVehicleData

Type Function
Sender SDL
Purpose Set RC module settings

RC.SetInteriorvehicledata represents a request from an application to change settings of requested RC module. This RPC can be sent to the HMI for an

application that is registered with REMOTE_CONTROL appHMIType and in one of the following states: FULL, LIMITTED, BACKGROUND. Module signed by the application in such request has to be available on HMI and allowed for control change settings

MUST

1. Modules sent by application must be available on HMI
2. Access to control module settings is defined by access mode entered by user on HMI
3. Module settings can be changed for settable parameters only
4. Requested module items have to be available for such module on HMI

REQUEST

PARAMETERS

NAME	TYPE	MANDATORY	ADDITIONAL	DESCRIPTION
moduleData	Common.ModuleData	true		The module type and data to set
appID	Integer	true		Internal SDL-assigned ID of the related application

RESPONSE

PARAMETERS

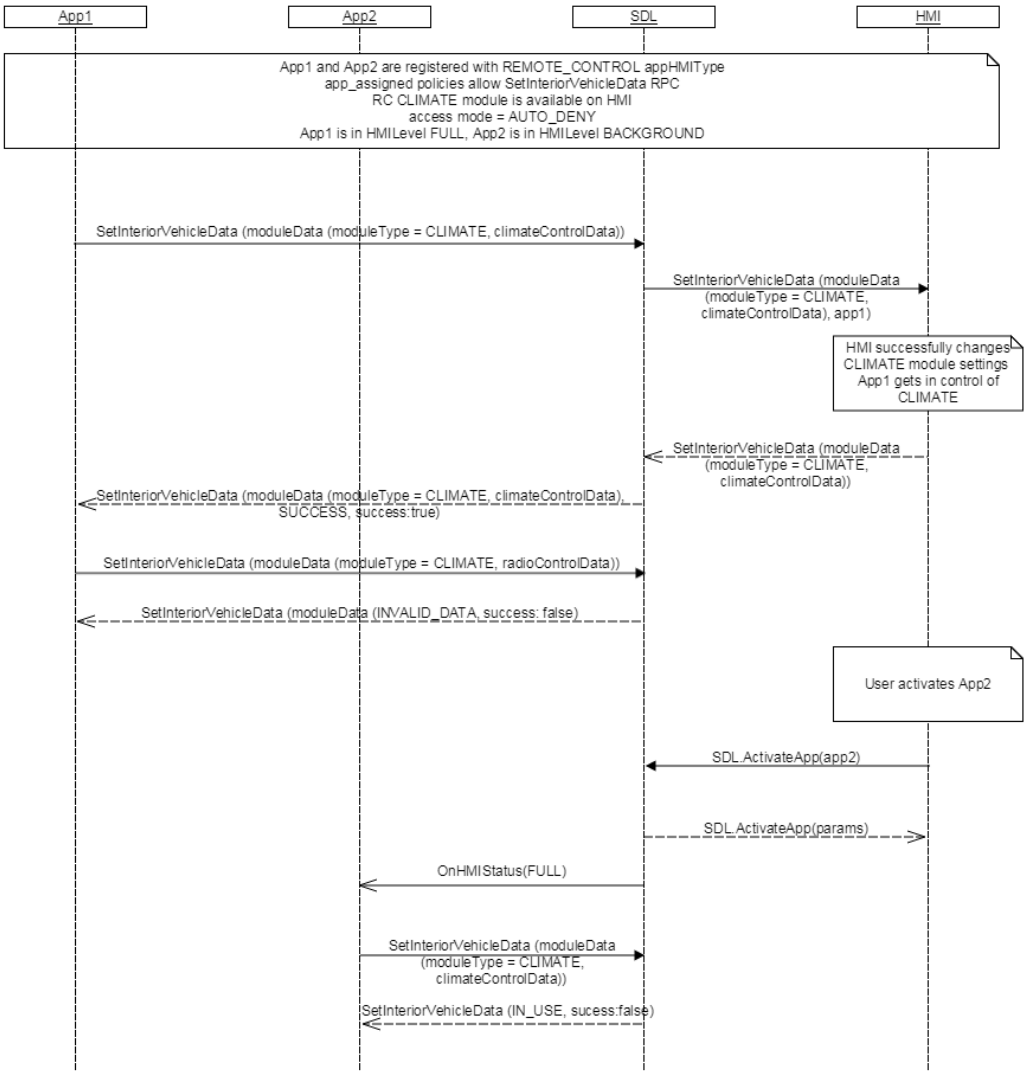
NAME	TYPE	MANDATORY	ADDITIONAL	DESCRIPTION
moduleData	Common.ModuleData	true		

Sequence Diagrams

SEQUENCE DIAGRAM

SetInteriorVehicleData

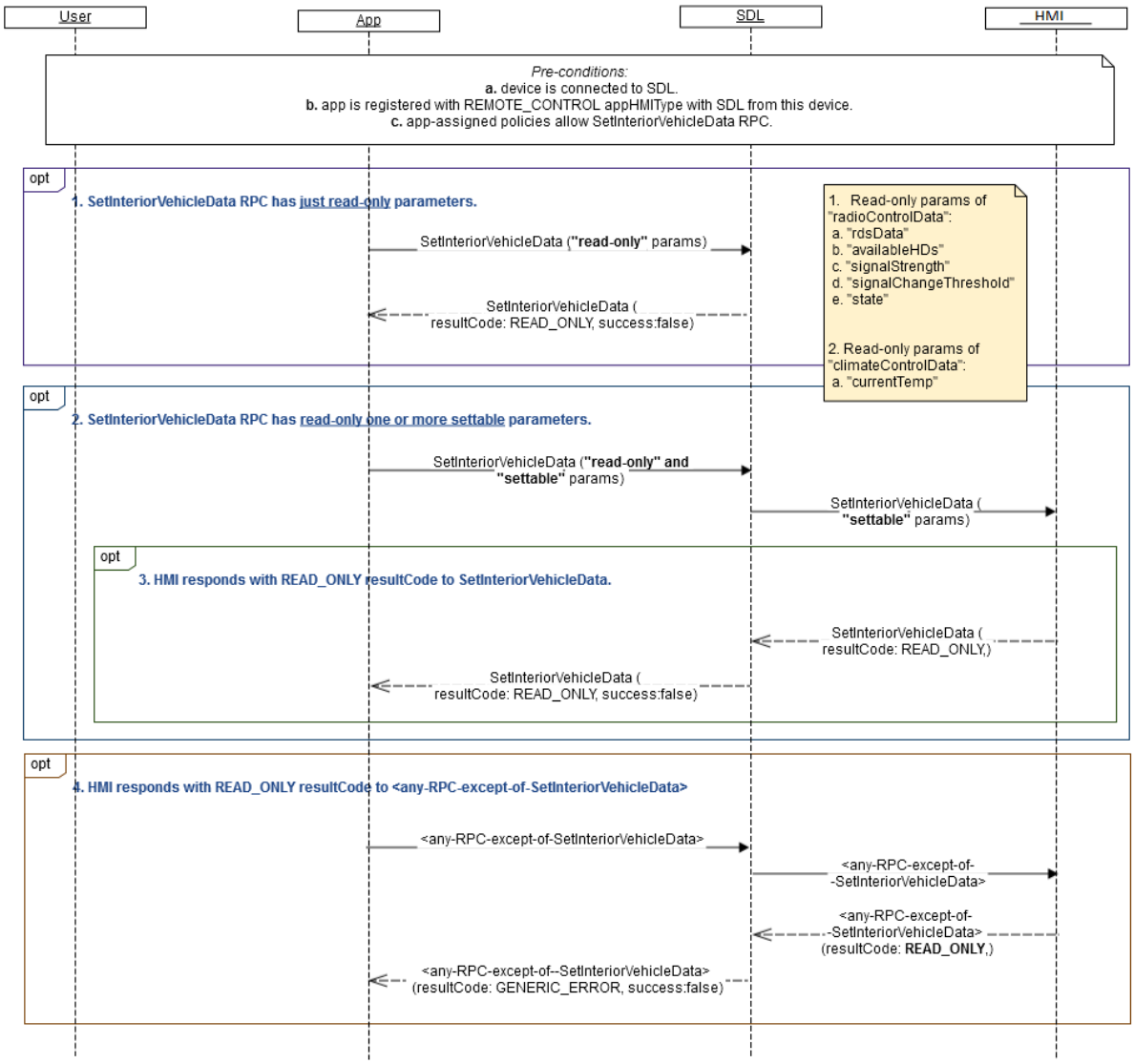
View Diagram



SEQUENCE DIAGRAM

SetInteriorvehicleData-READ_ONLY

View Diagram


User
App
SDL
HMI
Pre-conditions:
a. device is connected to SDL.
b. app is registered with REMOTE_CONTROL appHMIType with SDL from this device.
c. app-assigned policies allow SetInteriorVehicleData RPC.
opt
1. SetInteriorVehicleData RPC has just read-only parameters.
SetInteriorVehicleData ("read-only" params)
SetInteriorVehicleData (resultCode: READ_ONLY, success:false)
1. Read-only params of "radioControlData":
a. "rdsData"
b. "availableHDs"
c. "signalStrength"
d. "signalChangeThreshold"
e. "state"
2. Read-only params of "climateControlData":
a. "currentTemp"
opt
2. SetInteriorVehicleData RPC has read-only one or more settable parameters.
SetInteriorVehicleData ("read-only" and "settable" params)
SetInteriorVehicleData ("settable" params)
opt
3. HMI responds with READ_ONLY resultCode to SetInteriorVehicleData.
SetInteriorVehicleData (resultCode: READ_ONLY,)
SetInteriorVehicleData (resultCode: READ_ONLY, success:false)
opt
4. HMI responds with READ_ONLY resultCode to <any-RPC-except-of-SetInteriorVehicleData>
<any-RPC-except-of-SetInteriorVehicleData>
<any-RPC-except-of- -SetInteriorVehicleData>
<any-RPC-except-of- -SetInteriorVehicleData> (resultCode: READ_ONLY,)
<any-RPC-except-of--SetInteriorVehicleData> (resultCode: GENERIC_ERROR, success:false)

Example Request


```
{
   "id": 31,
   "jsonrpc": "2.0",
   "method": "RC.SetInteriorVehicleData",
   "params": {
      "appID": 306873852,
      "moduleData": {
         "climateControlData": {
            "acEnable": true,
            "acMaxEnable": true,
            "autoModeEnable": true,
            "circulateAirEnable": true,
            "currentTemperature": {
               "unit": "FAHRENHEIT",
               "value": 20.1
            },
            "defrostZone": "FRONT",
            "desiredTemperature": {
               "unit": "CELSIUS",
               "value": 10.5
            },
            "dualModeEnable": true,
            "fanSpeed": 50,
            "ventilationMode": "BOTH"
         },
         "moduleType": "CLIMATE"
      }
   }
}
```

Example Response


```
{
   "id": 31,
   "jsonrpc": "2.0",
   "result": {
      "code": 0,
      "method": "RC.SetInteriorVehicleData",
      "moduleData": {
         "climateControlData": {
            "acEnable": true,
            "acMaxEnable": true,
            "autoModeEnable": true,
            "circulateAirEnable": true,
            "currentTemperature": {
               "unit": "FAHRENHEIT",
               "value": 20.1
            },
            "defrostZone": "FRONT",
            "desiredTemperature": {
               "unit": "CELSIUS",
               "value": 10.5
            },
            "dualModeEnable": true,
            "fanSpeed": 50,
            "ventilationMode": "BOTH"
         },
         "moduleType": "CLIMATE"
      }
   }
}
```

Example Error

```
{
   "error": {
      "code": 26,
      "data": {
         "method": "RC.SetInteriorVehicleData"
      },
      "message": "Read only parameters received"
   },
   "id": 31,
   "jsonrpc": "2.0"
}
```

OnRCStatus

Type Notification
Sender SDL
Purpose Inform HMI and mobile application about changes in RC module allocation.

Notification

SDL must send OnRCStatus notification when:
* RC application registers with SDL
* A module is allocated to an application
* A module is de-allocated from an application (the module is freed when application exits, disconnects, goes to HMI level NONE)
* An application no longer has the right to control the module (for example, a policy update can revoke the application's access to a module)

PARAMETERS

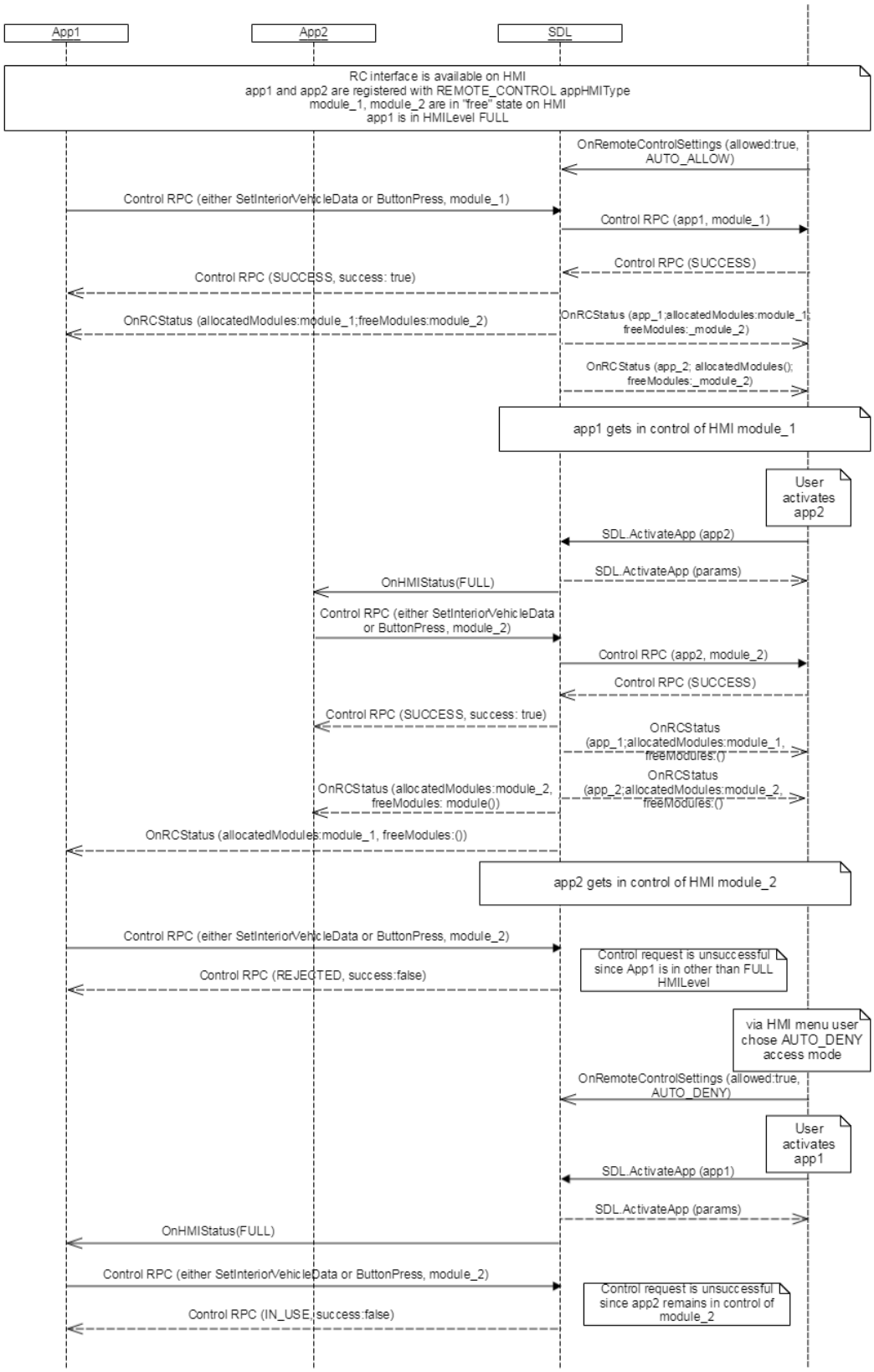
NAME	TYPE	MANDATORY	ADDITIONAL
appID	Integer	true	ID of selected application
allocatedModules	Common.ModuleData	true	Contains a list (zero or more) of module types that are allocated to the application
freeModules	Common.ModuleData	true	Contains a list (zero or more) of module types that are free to access for the application

Sequence Diagrams

SEQUENCE DIAGRAM

OnRCStatus

View Diagram


App1
App2
SDL
RC interface is available on HMI
app1 and app2 are registered with REMOTE_CONTROL appHMIType
module_1, module_2 are in "free" state on HMI
app1 is in HMILevel FULL
OnRemoteControlSettings (allowed:true, AUTO_ALLOW)
Control RPC (either SetInteriorVehicleData or ButtonPress, module_1)
Control RPC (app1, module_1)
Control RPC (SUCCESS)
Control RPC (SUCCESS, success: true)
OnRCStatus (allocatedModules:module_1;freeModules:module_2)
OnRCStatus (app_1;allocatedModules:module_1; freeModules:_module_2)
OnRCStatus (app_2; allocatedModules(); freeModules:_module_2)
app1 gets in control of HMI module_1
User activates app2
SDL.ActivateApp (app2)
SDL.ActivateApp (params)
OnHMIStatus(FULL)
Control RPC (either SetInteriorVehicleData or ButtonPress, module_2)
Control RPC (app2, module_2)
Control RPC (SUCCESS)
Control RPC (SUCCESS, success: true)
OnRCStatus (app_1;allocatedModules:module_1, freeModules:())
OnRCStatus (app_2;allocatedModules:module_2, freeModules:())
OnRCStatus (allocatedModules:module_2, freeModules: module())
OnRCStatus (allocatedModules:module_1, freeModules:())
app2 gets in control of HMI module_2
Control RPC (either SetInteriorVehicleData or ButtonPress, module_2)
Control request is unsuccessful since App1 is in other than FULL HMILevel
Control RPC (REJECTED, success:false)
via HMI menu user chose AUTO_DENY access mode
OnRemoteControlSettings (allowed:true, AUTO_DENY)
User activates app1
SDL.ActivateApp (app1)
SDL.ActivateApp (params)
OnHMIStatus(FULL)
Control RPC (either SetInteriorVehicleData or ButtonPress, module_2)
Control request is unsuccessful since app2 remains in control of module_2
Control RPC (IN_USE, success:false)

JSON EXAMPLE NOTIFICATION

```json
{
"jsonrpc": "2.0",
"method": "RC.OnRCStatus",
"params": {
"allocatedModules" : [],
"freeModules" : [
{
"moduleType" : "CLIMATE"
},
{
"moduleType" : "RADIO"
},
{
"moduleType" : "AUDIO"
},
{
"moduleType" : "LIGHT"
},
{
"moduleType" : "HMI_SETTINGS"
},
{
"moduleType" : "SEAT"
}
]
}
```

PublishAppService

Type

Function

Sender

HMI

Purpose

Registers a service offered by the HMI on the module

REQUEST

PARAMETERS

NAME	TYPE	MANDATORY	ADDITIONAL
appServiceManifest	Common.AppServiceManifest	true	

RESPONSE

PARAMETERS

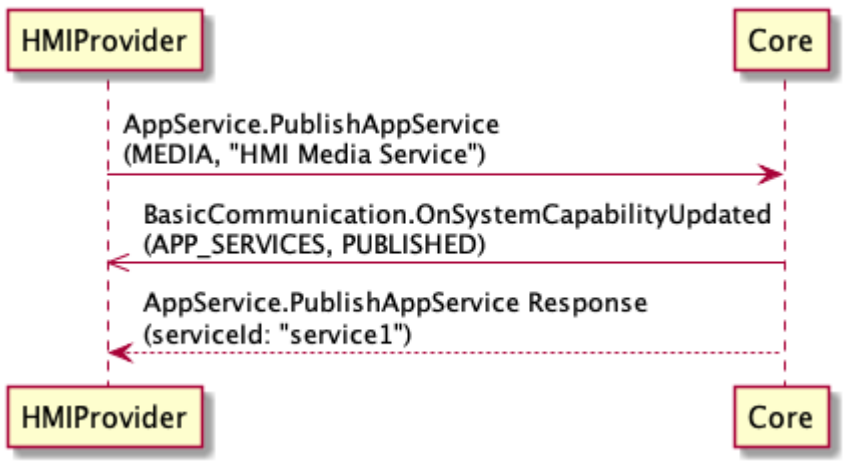
NAME	TYPE	MANDATORY	ADDITIONAL
appServiceRecord	Common.AppServiceRecord	false	

Sequence Diagrams

SEQUENCE DIAGRAM

PublishAppService

View Diagram



Example Request

```
{
   "id": 1000,
   "jsonrpc": "2.0",
   "method": "AppService.PublishAppService",
   "params": {
      "appServiceManifest": {
         "allowAppConsumers": true,
         "handledRPCs": [41],
         "rpcSpecVersion": {
            "majorVersion": 5,
            "minorVersion": 1
         },
         "serviceName": "IVI_Media",
         "serviceType": "MEDIA"
      }
   }
}
```

Example Response


```
{
  "id" : 1000,
  "jsonrpc" : "2.0",
  "result" : {
    "appServiceRecord" : {
      "serviceActive" : true,
      "serviceID" :
"1f89547cc0b12d5a52e896c45e141497f7af50e1e2dc8705914e75ef6fbe
",
      "serviceManifest" : {
        "allowAppConsumers" : true,
        "handledRPCs" : [ 41 ],
        "rpcSpecVersion" : {
          "majorVersion" : 5,
          "minorVersion" : 1
        },
        "serviceName" : "IVI_Media",
        "serviceType" : "MEDIA"
      },
      "servicePublished" : true
    },
    "code" : 0,
    "method" : "AppService.PublishAppService"
  }
}
```


GetAppServiceData

Type

Request

Sender

HMI / SDL

Purpose

Requests app service data from the active service of a given type. Also gives the option to subscribe to future app service data updates for this serviceType

NOTE

- **HMI->SDL** if HMI App Service Consumer (ASC) is requesting service data
- **SDL->HMI** if HMI App Service Provider (ASP) is receiving a service data request

REQUEST

PARAMETERS

NAME	TYPE	MANDATORY	ADDITIONAL
serviceType	String	true	
subscribe	Boolean	false	

RESPONSE

PARAMETERS

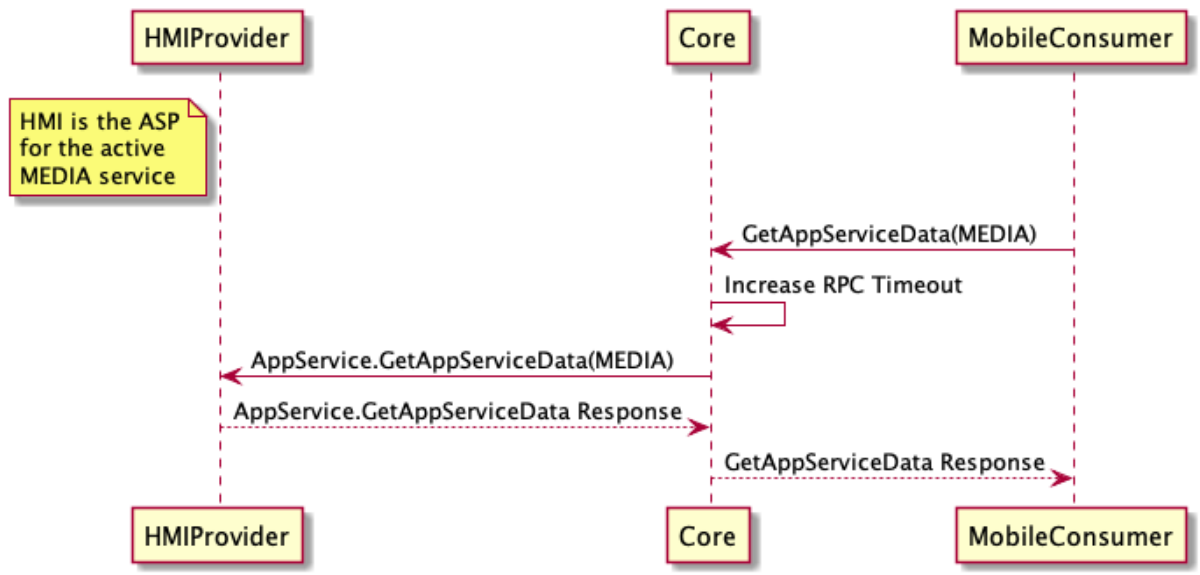
NAME	TYPE	MANDATORY	ADDITIONAL
serviceData	Common.AppServiceData	false	

Sequence Diagrams

SEQUENCE DIAGRAM

GetAppServiceData (HMI Provider)

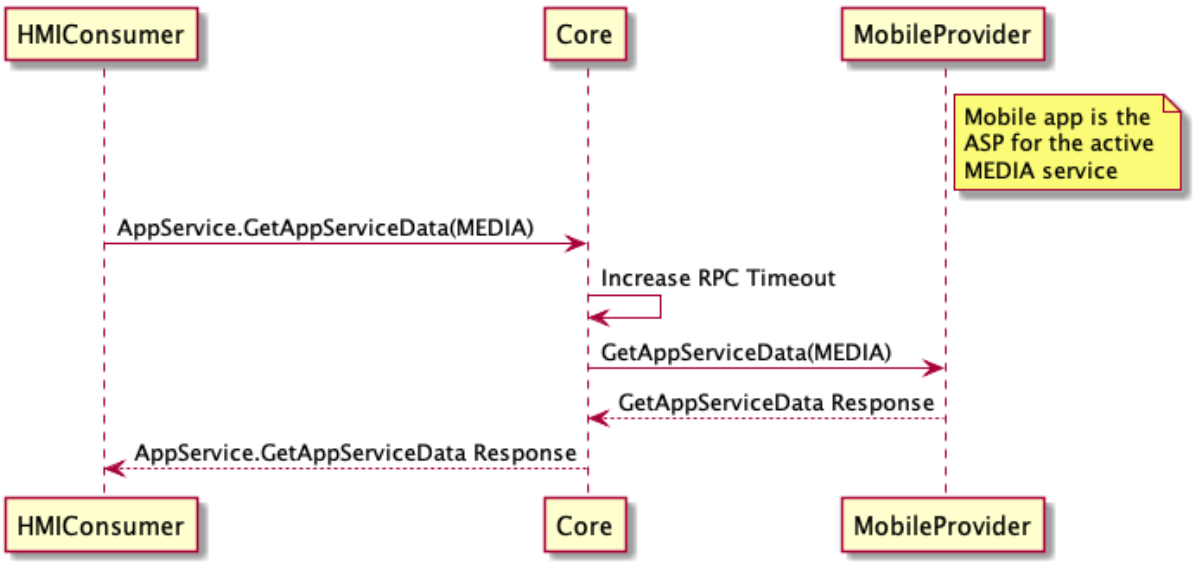
View Diagram



SEQUENCE DIAGRAM

GetAppServiceData (HMI Consumer)

View Diagram



Example Request

```
{
  "id": 1000,
  "jsonrpc": "2.0",
  "method": "AppService.GetAppServiceData",
  "params": {
    "serviceType": "MEDIA"
  }
}
```

Example Response


```
{
  "id": 1000,
  "jsonrpc": "2.0",
  "result": {
    "code": 0,
    "method": "AppService.GetAppServiceData",
    "resultCode": 0,
    "serviceData": {
      "mediaServiceData": {
        "isExplicit": false,
        "mediaAlbum": "Book Name",
        "mediaArtist": "Author name",
        "mediaTitle": "Chapter name",
        "mediaType": "AUDIOBOOK",
        "queueCurrentTrackNumber": 12,
        "queuePlaybackDuration": 4000,
        "queuePlaybackProgress": 2200,
        "queueTotalTrackCount": 25,
        "trackPlaybackDuration": 300,
        "trackPlaybackProgress": 200
      },
      "serviceID":
"9c6697b90f561cc599af19f81e9cf68a6848d6df1cdd63820d75ebfd7c72
",
      "serviceType": "MEDIA"
    },
    "success": true
  }
}
```

Example Error


```
{
  "error": {
    "code": 9,
    "data": {
      "method": "AppService.GetAppServiceData"
    },
    "message": "No app service provider with serviceType: MUSIC is
available"
  },
  "id": 1000,
  "jsonrpc": "2.0"
}
```


OnAppServiceData

Type Notification
Sender HMI / SDL
Purpose Sent by the app service provider when its service data is updated. Broadcast by SDL to all consumers subscribed to this service type.

NOTE

- **HMI->SDL** if HMI App Service Provider (ASP) is sending updated service data
- **SDL->HMI** if HMI App Service Consumer (ASC) is receiving updated service data

Notification

PARAMETERS

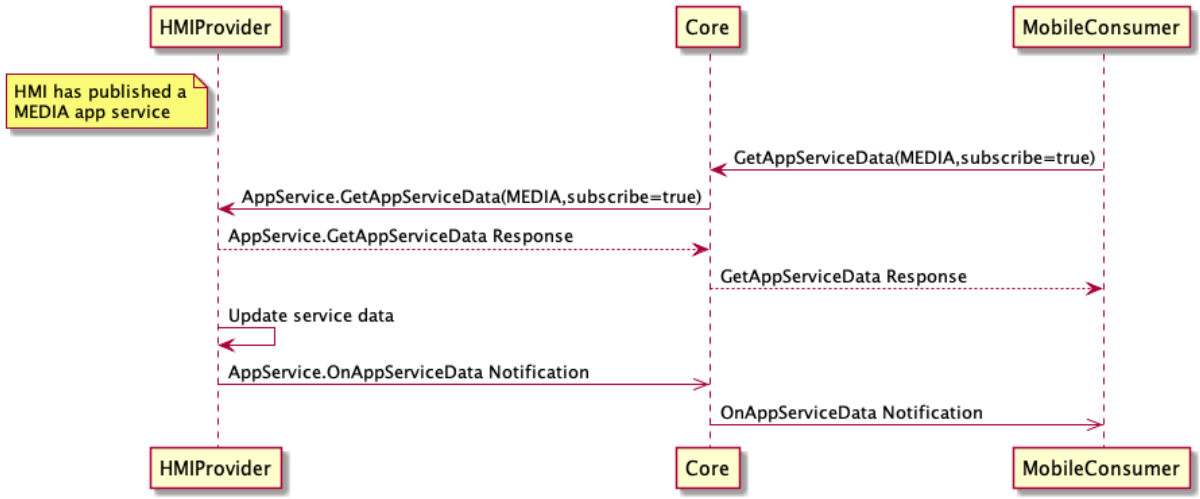
NAME	TYPE	MANDATORY	ADDITIONAL
serviceData	Common.AppServiceData	true	

Sequence Diagrams

SEQUENCE DIAGRAM

OnAppServiceData (HMI Provider)

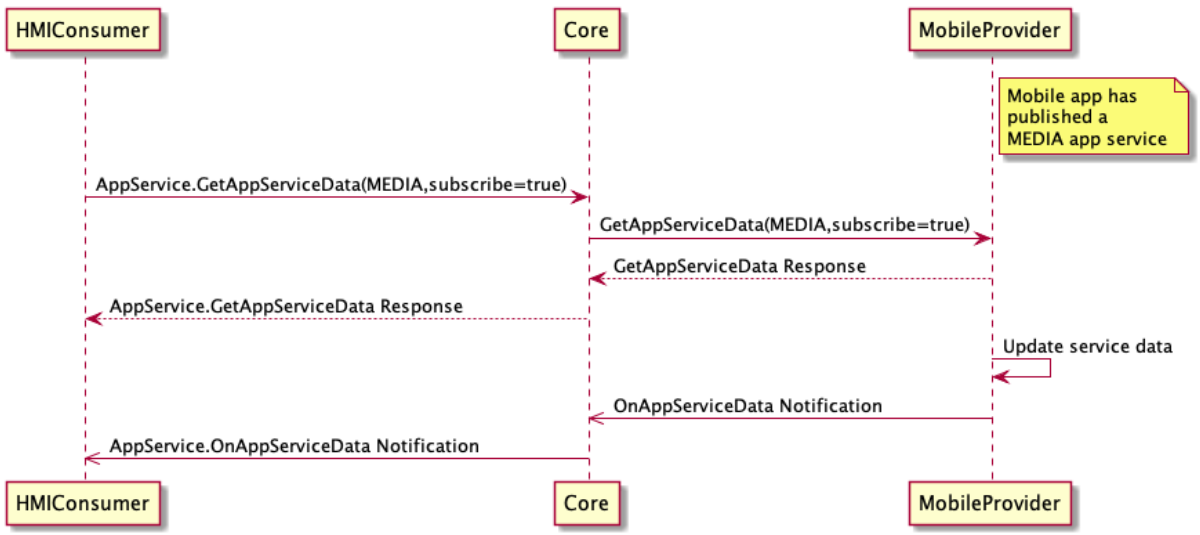
View Diagram



SEQUENCE DIAGRAM

OnAppServiceData (HMI Consumer)

View Diagram



Example Notification


```
{
 "jsonrpc": "2.0",
 "method": "AppService.OnAppServiceData",
 "params": {
  "serviceData": {
   "mediaServiceData": {
    "isExplicit": false,
    "mediaAlbum": "Book Name",
    "mediaArtist": "Author name",
    "mediaTitle": "Chapter name",
    "mediaType": "AUDIOBOOK",
    "queueCurrentTrackNumber": 12,
    "queuePlaybackDuration": 4000,
    "queuePlaybackProgress": 2200,
    "queueTotalTrackCount": 25,
    "trackPlaybackDuration": 300,
    "trackPlaybackProgress": 200
   },
   "serviceID":
"9c6697b90f561cc599af19f81e9cf68a6848d6df1cdd63820d75ebfd7c72
",
   "serviceType": "MEDIA"
  }
 }
}
```


PerformAppServiceInteraction

Type
Function

Sender

SDL / HMI

Purpose

Request a service provider to perform a predefined action.

REQUEST

MUST

Process `serviceUri` according to whatever predefined schema is offered by the HMI service

NOTE

If the HMI is the App Service Consumer (ASC), SDL will:
1. Activate the service specified with `serviceID` if `requestServiceActive` is set to `true`

PARAMETERS

NAME	TYPE	MANDATORY	ADDITIONAL
serviceUri	String	true	
serviceID	String	true	
originApp	String	false	Non-mandatory only if HMI is the ASC
requestServiceActive	Boolean	false	

RESPONSE

PARAMETERS

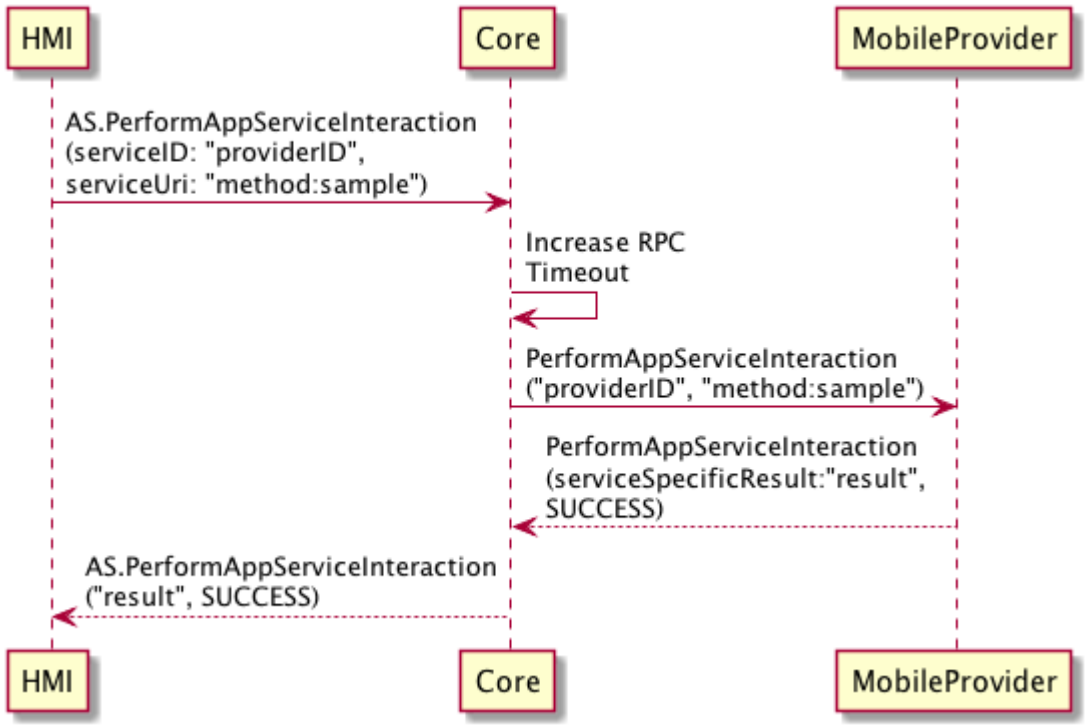
NAME	TYPE	MANDATORY	ADDITIONAL
serviceSpecificResult	String	false	

Sequence Diagrams

SEQUENCE DIAGRAM

PerformAppServiceInteraction with HMI ASC

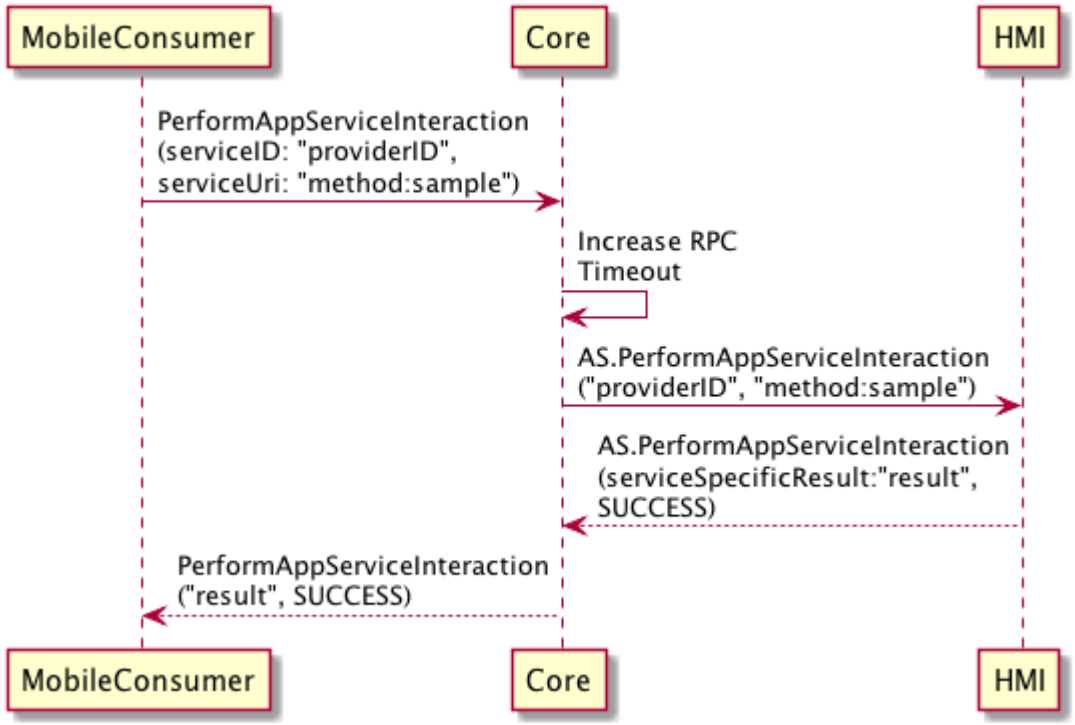
View Diagram



SEQUENCE DIAGRAM

PerformAppServiceInteraction with Mobile ASC

View Diagram



Example Request

```
{
  "id": 1000,
  "jsonrpc": "2.0",
  "method": "AppService.PerformAppServiceInteraction",
  "params": {
    "serviceUri": "host:sample.service.uri",
    "serviceID": "service_id",
    "originApp": "123456"
  }
}
```

Example Response


```
{
  "id" : 1000,
  "jsonrpc" : "2.0",
  "result" : {
    "serviceSpecificResult" : "QUEUED",
    "code" : 0,
    "method" : "AppService.PerformAppServiceInteraction"
  }
}
```


Example Error


```
{
  "id" : 1000,
  "jsonrpc" : "2.0",
  "result" : {
    "code" : 1,
    "message" : "No known service interaction matching URI",
    "data" : {
       "method" : "AppService.PerformAppServiceInteraction"
    }
  }
}
```


GetActiveServiceConsent

Type

Function

Sender

SDL

Purpose

Request user consent to activate an app service

REQUEST

MUST

1. Prompt user to activate service with `serviceID`
2. Respond with the result from the user in the `activate` parameter

PARAMETERS

NAME	TYPE	MANDATORY	ADDITIONAL
serviceID	String	true	

RESPONSE

PARAMETERS

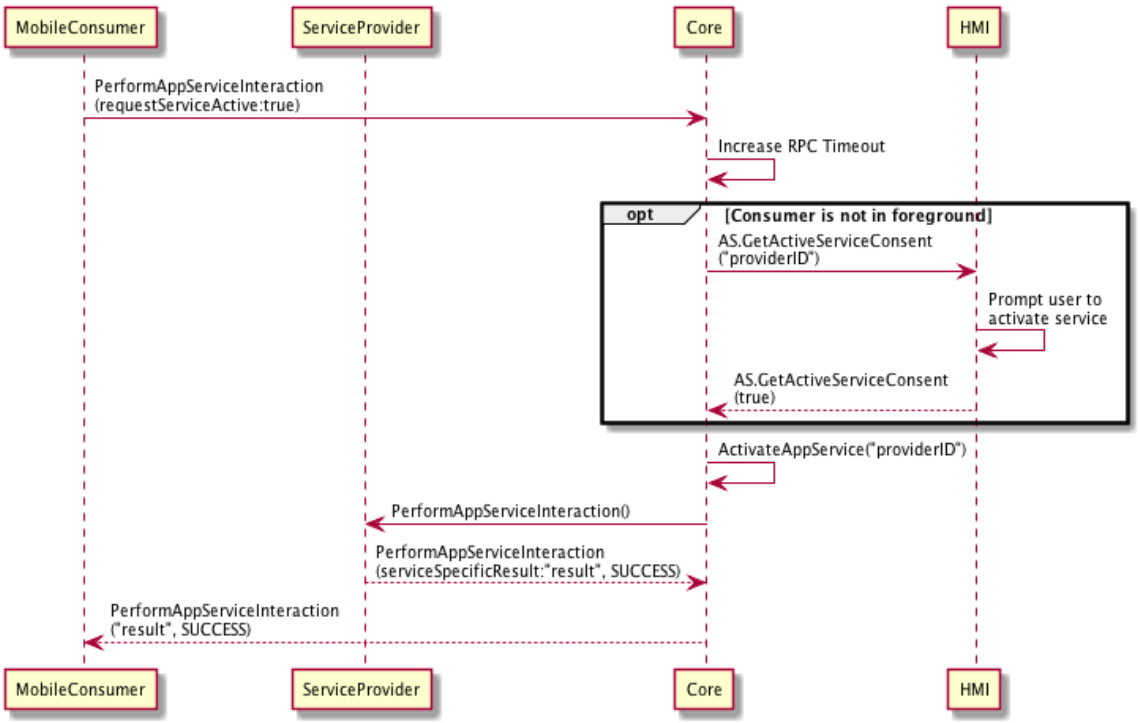
NAME	TYPE	MANDATORY	ADDITIONAL
activate	Boolean	true	

Sequence Diagrams

SEQUENCE DIAGRAM

GetActiveServiceConsent (user accepts prompt)

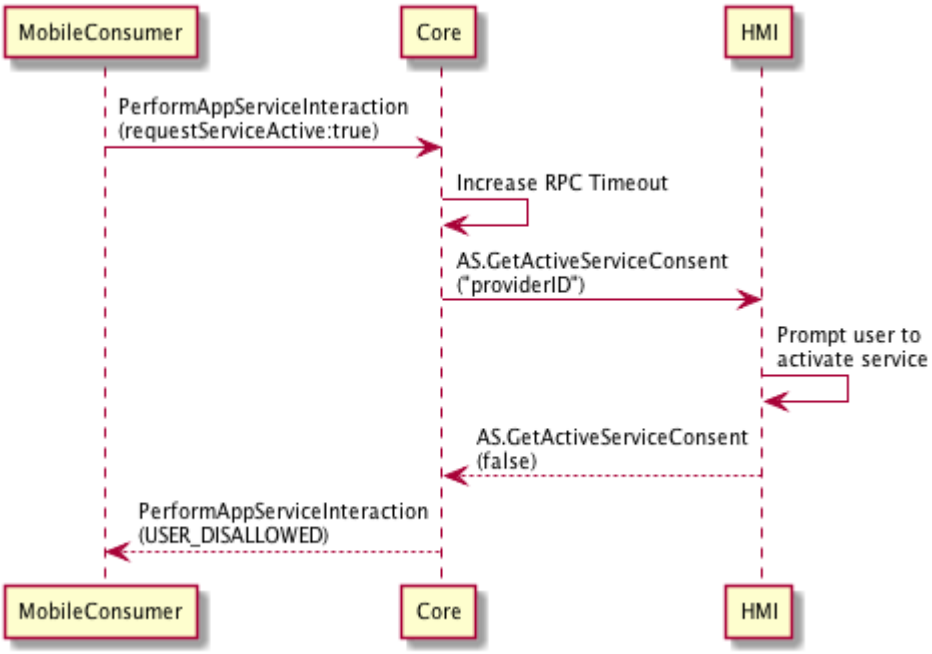
View Diagram



SEQUENCE DIAGRAM

GetActiveServiceConsent (user rejects prompt)

View Diagram



Example Request

```
{
  "id": 1000,
  "jsonrpc": "2.0",
  "method": "AppService.GetActiveServiceConsent",
  "params": {
    "serviceID": "service_id"
  }
}
```

Example Response


```
{
  "id" : 1000,
  "jsonrpc" : "2.0",
  "result" : {
    "activate" : true,
    "code" : 0,
    "method" : "AppService.GetActiveServiceConsent"
  }
}
```


Example Error


```
{
  "id" : 1000,
  "jsonrpc" : "2.0",
  "result" : {
    "code" : 13,
    "message" : "No known service with given ID",
    "data" : {
       "method" : "AppService.GetActiveServiceConsent"
    }
  }
}
```


GetAppServiceRecords

Type

Request

Sender

HMI

Purpose

Get service records of a specific service type.

NOTE

If `serviceType` is not included in request, all service records will be returned

REQUEST

PARAMETERS

NAME	TYPE	MANDATORY	ADDITIONAL
serviceType	String	false	

RESPONSE

PARAMETERS

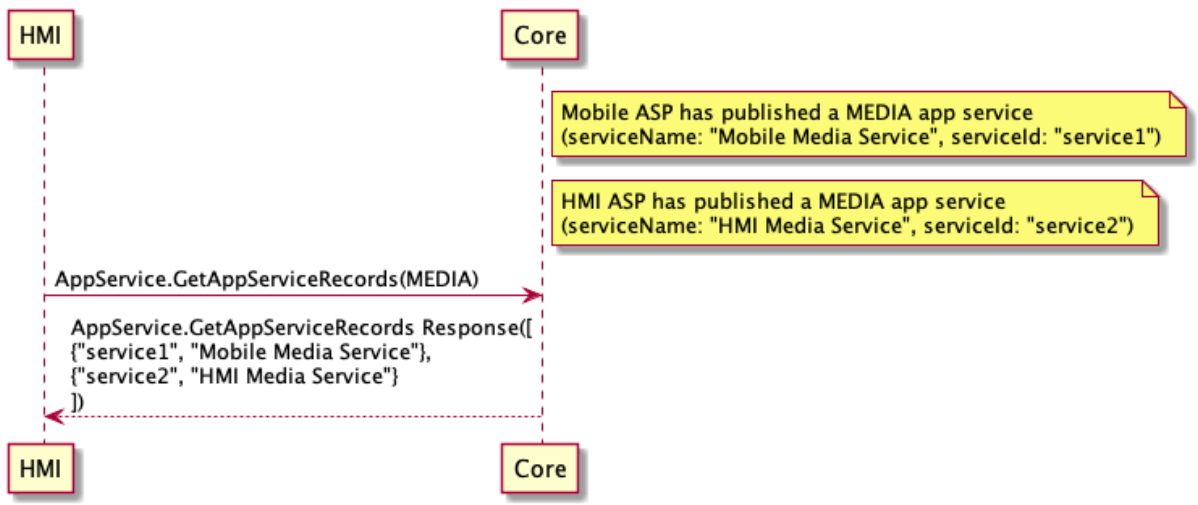
NAME	TYPE	MANDATORY	ADDITIONAL
serviceRecords	Common.AppServiceRecord	true	array: true

Sequence Diagrams

SEQUENCE DIAGRAM

GetAppServiceRecords

View Diagram



Example Request

```
{
  "id": 1003,
  "jsonrpc": "2.0",
  "method": "AppService.GetAppServiceRecords",
  "params": {
    "serviceType": "MEDIA"
  }
}
```

Example Response


```
{
  "id": 1003,
  "jsonrpc": "2.0",
  "result": {
    "code": 0,
    "method": "AppService.GetAppServiceRecords",
    "serviceRecords": [
      {
        "serviceActive": false,
        "serviceID":
"c9503a4f983e3dd31a7a14564d405cdf84769c9b9a71cae9cc211a0b74(
,
        "serviceManifest": {
          "allowAppConsumers": true,
          "rpcSpecVersion": {
            "majorVersion": 5,
            "minorVersion": 1,
            "patchVersion": 0
          },
          "serviceName": "Mobile Media Service",
          "serviceType": "MEDIA"
        },
        "servicePublished": true
      },
      {
        "serviceActive": true,
        "serviceID":
"d6d8d2fb2bfcc00033804fb19e9fb7d6070d2c166f49881563276f17478(
,
        "serviceManifest": {
          "allowAppConsumers": true,
          "rpcSpecVersion": {
            "majorVersion": 5,
            "minorVersion": 1,
            "patchVersion": 0
          },
          "serviceName": "Waze Music",
          "serviceType": "MEDIA"
        },
        "servicePublished": true
      }
    ]
  }
}
```

AppServiceActivation

Type Function
Sender HMI
Purpose

Request to activate/deactivate a given service, as well as set a service to the default

REQUEST

NOTE

SDL will:
1. Activate the service with ID `serviceID` if `activate` is set to true and the service is currently inactive. This will deactivate the current active service of its type.
2. Deactivate the service with ID `serviceID` if `activate` is set to false and the service is currently active. If the App Service Consumer (ASC) is a mobile application, the embedded service for this type will be activated in its place if available.
3. Make the service with ID `serviceID` the default service for its type if `setAsDefault` is set to true. This will replace the existing default service for this service type.
4. Unset the service with ID `serviceID` as the default service for its service type if `setAsDefault` is set to false and this service is currently the default.

PARAMETERS

NAME	TYPE	MANDATORY	ADDITIONAL
serviceID	String	true	
activate	Boolean	true	
setAsDefault	Boolean	false	

RESPONSE

PARAMETERS

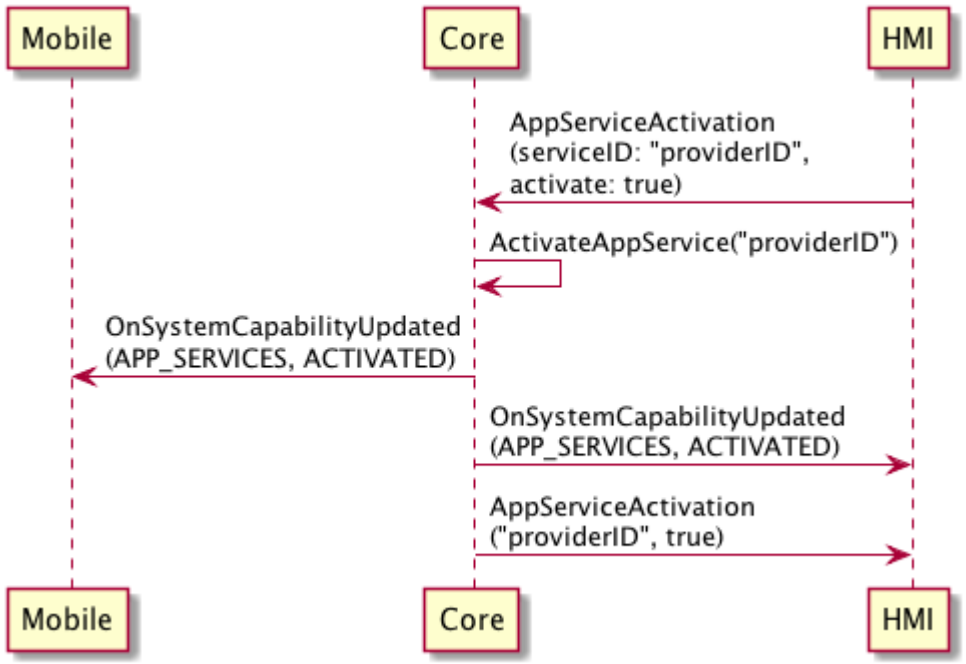
NAME	TYPE	MANDATORY	ADDITIONAL
serviceID	String	true	
activate	Boolean	true	
setAsDefault	Boolean	false	

Sequence Diagrams

SEQUENCE DIAGRAM

AppServiceActivation activate service

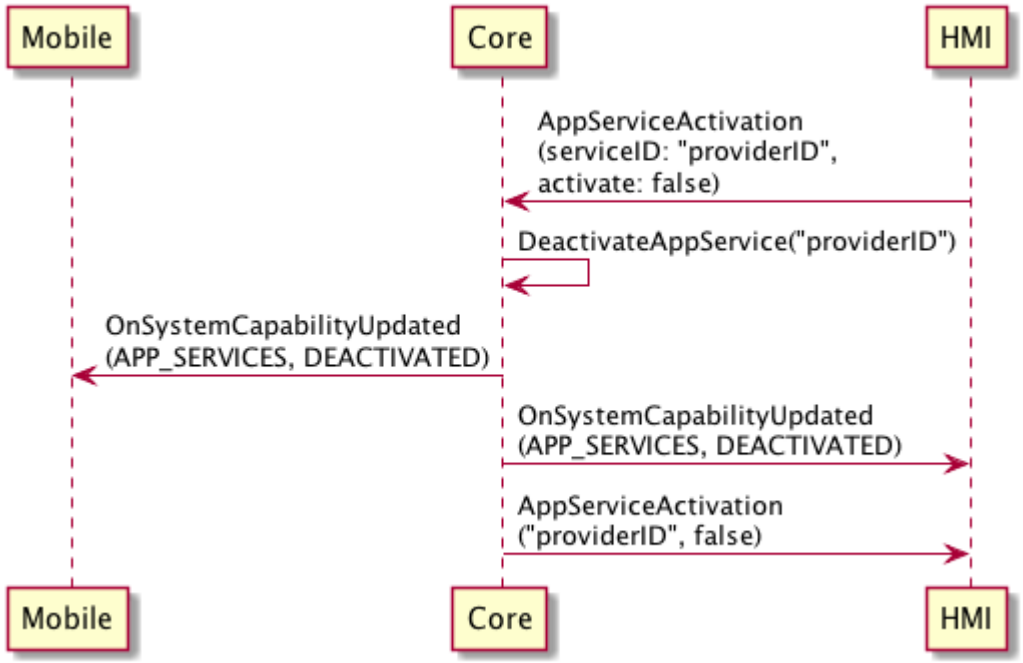
View Diagram



SEQUENCE DIAGRAM

AppServiceActivation deactivate service

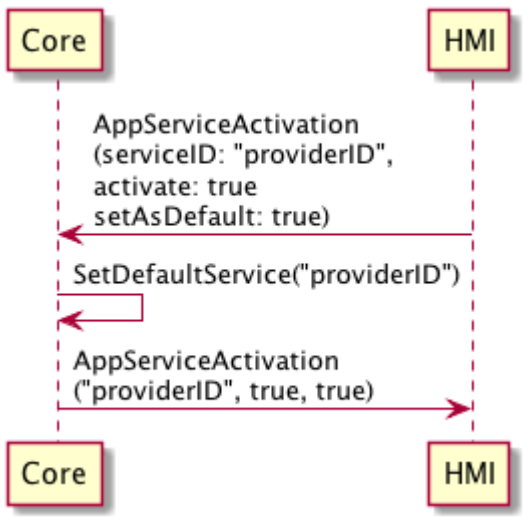
View Diagram



SEQUENCE DIAGRAM

AppServiceActivation set default service

View Diagram



Example Request

```
{
  "id": 1000,
  "jsonrpc": "2.0",
  "method": "AppService.AppServiceActivation",
  "params": {
    "serviceID": "service_id",
    "activate": true,
    "setAsDefault": true
  }
}
```

Example Response

```
{
  "id" : 1000,
  "jsonrpc" : "2.0",
  "result" : {
    "serviceID": "service_id",
    "activate": true,
    "setAsDefault": true,
    "code" : 0,
    "method" : "AppService.AppServiceActivation"
  }
}
```

Example Error

```
{
  "id" : 1000,
  "jsonrpc" : "2.0",
  "result" : {
    "code" : 13,
    "message" : "No known service with given ID",
    "data" : {
       "method" : "AppService.AppServiceActivation"
    }
  }
}
```

Enumerations

KeyboardEvent

NAME	VALUE	DESCRIPTION
KEYPRESS	0	
ENTRY_SUBMITTED	1	
ENTRY_VOICE	2	
ENTRY_CANCELLED	3	
ENTRY_ABORTED	4	

FuelCutoffStatus

NAME	VALUE	DESCRIPTION
TERMINATE_FUEL	0	
NORMAL_OPERATION	1	
FAULT	2	

TransportType

NAME	VALUE	DESCRIPTION
BLUETOOTH	0	
USB_IOS	1	
USB_AOA	2	
WIFI	3	
CLOUD_WEBSOCKET	4	Websocket connection used to connect to remote cloud application

EventTypes

NAME	VALUE	DESCRIPTION
PHONE_CALL	0	Phone call is active
EMERGENCY_EVENT	1	Active emergency event, active parking event
DEACTIVATE_HMI	2	GAL/DIO is active
AUDIO_SOURCE	3	Navigated to audio (radio, etc)
EMBEDDED_NAVI	4	Navigated to navigation screen

ButtonPressMode

NAME	VALUE	DESCRIPTION
LONG	0	
SHORT	1	

VRCommandType

NAME	VALUE	DESCRIPTION
Choice	0	
Command	1	

AppHMIType

NAME	VALUE	DESCRIPTION
DEFAULT	0	
COMMUNICATION	1	
MEDIA	2	
MESSAGING	3	
NAVIGATION	4	
INFORMATION	5	
SOCIAL	6	
BACKGROUND_PROCESS	7	
TESTING	8	
SYSTEM	9	
PROJECTION	10	
REMOTE_CONTROL	11	

CloudConnectionStatus

NAME	VALUE	DESCRIPTION
NOT_CONNECTED	0	No active websocket session or ongoing connection attempts
CONNECTED	1	Websocket is active
RETRY	2	Websocket connection failed and retry attempts are ongoing

WayPointType

NAME	VALUE	DESCRIPTION
ALL	0	All waypoints including destination
DESTINATION	1	Only destination

SpeechCapabilities

NAME	VALUE	DESCRIPTION
TEXT	0	Uses plain text for performing TTS
SAPI_PHONEMES	1	Uses the Speech API Phoneme representation of a phrase for performing TTS
LHPLUS_PHONEMES	2	Uses the LH+ Phoneme representation of a phrase for performing TTS
PRE_RECORDED	3	
SILENCE	4	
FILE	5	Uses an audio file sent to SDL via a PutFile RPC to perform TTS or play generic sounds in conjunction with TTS

TextFieldName

NAME	VALUE	DESCRIPTION
mainField1	0	
mainField2	1	
mainField3	2	
mainField4	3	
statusBar	4	
mediaClock	5	
mediaTrack	6	
alertText1	7	
alertText2	8	
alertText3	9	
scrollableMessageBody	10	
initialInteractionText	11	
navigationText1	12	
navigationText2	13	
ETA	14	
totalDistance	15	
audioPassThruDisplayText1	16	
audioPassThruDisplayText2	17	

NAME	VALUE	DESCRIPTION
sliderHeader	18	
sliderFooter	19	
menuName	20	
secondaryText	21	
tertiaryText	22	
menuTitle	23	
navigationText	24	
notificationText	25	
locationName	26	
locationDescription	27	
addressLines	28	
phoneNumber	29	
timeToDestination	30	
turnText	31	

MetadataType

NAME	VALUE	DESCRIPTION
mediaTitle	0	This field contains the title of the current audio track.
mediaArtist	1	This field contains the artist/creator of the current audio track.
mediaAlbum	2	This field contains the album title of the current audio track.
mediaYear	3	This field contains the creation year of the current audio track.
mediaGenre	4	This field contains the genre of the current audio track.
mediaStation	5	This field contains the name of the current media source.
rating	6	This field contains a rating of some form.
currentTemperature	7	This field contains the current temperature.
maximumTemperature	8	This field contains the maximum temperature for the day.
minimumTemperature	9	This field contains the minimum temperature for the day.
weatherTerm	10	This field contains the current weather (cloudy, clear, etc.).
humidity	11	This field contains the current humidity value.

AmbientLightStatus

NAME	VALUE	DESCRIPTION
NIGHT	0	
TWILIGHT_1	1	
TWILIGHT_2	2	
TWILIGHT_3	3	
TWILIGHT_4	4	
DAY	5	
UNKNOWN	6	
INVALID	7	

StatisticsType

NAME	VALUE	DESCRIPTION
iAPP_BUFFER_FULL	0	

VehicleDataEventStatus

NAME	VALUE	DESCRIPTION
NO_EVENT	0	
NO	1	
YES	2	
NOT_SUPPORTED	3	
FAULT	4	

KeyboardLayout

NAME	VALUE	DESCRIPTION
QWERTY	0	
QWERTZ	1	
AZERTY	2	

ButtonEventMode

NAME	VALUE	DESCRIPTION
BUTTONUP	0	
BUTTONDOWN	1	

VrCapabilities

NAME	VALUE	DESCRIPTION
TEXT	0	

TextAlignment

NAME	VALUE	DESCRIPTION
LEFT_ALIGNED	0	
RIGHT_ALIGNED	1	
CENTERED	2	

DeviceState

NAME	VALUE	DESCRIPTION
UNKNOWN	0	
UNPAIRED	1	

SystemContext

NAME	VALUE	DESCRIPTION
MAIN	0	
VRSESSION	1	
MENU	2	
HMI_OBSCURED	3	
ALERT	4	

SoftButtonType

NAME	VALUE	DESCRIPTION
TEXT	0	
IMAGE	1	
BOTH	2	

DriverDistractionState

NAME	VALUE	DESCRIPTION
DD_ON	0	
DD_OFF	1	

MethodName

NAME	VALUE	DESCRIPTION
ALERT	0	
SPEAK	1	
AUDIO_PASS_THRU	2	
ALERT_MANEUVER	3	

CharacterSet

NAME	VALUE	DESCRIPTION
TYPE2SET	0	
TYPE5SET	1	
CID1SET	2	
CID2SET	3	

SamplingRate

NAME	VALUE	DESCRIPTION
8KHZ	0	
16KHZ	1	
22KHZ	2	
44KHZ	3	

PrerecordedSpeech

NAME	VALUE	DESCRIPTION
HELP_JINGLE	0	
INITIAL_JINGLE	1	
LISTEN_JINGLE	2	
POSITIVE_JINGLE	3	
NEGATIVE_JINGLE	4	

MediaClockFormat

NAME	VALUE	DESCRIPTION
CLOCK1	0	
CLOCK2	1	
CLOCK3	2	
CLOCKTEXT1	3	
CLOCKTEXT2	4	
CLOCKTEXT3	5	
CLOCKTEXT4	6	

TouchType

NAME	VALUE	DESCRIPTION
BEGIN	0	
MOVE	1	
END	2	
CANCEL	3	

HmiZoneCapabilities

NAME	VALUE	DESCRIPTION
FRONT	0	
BACK	1	

AlertType

NAME	VALUE	DESCRIPTION
UI	0	
BOTH	1	

ECallConfirmationStatus

NAME	VALUE	DESCRIPTION
NORMAL	0	
CALL_IN_PROGRESS	1	
CALL_CANCELLED	2	
CALL_COMPLETED	3	
CALL_UNSUCCESSFUL	4	
ECALL_CONFIGURED_OFF	5	
CALL_COMPLETE_DTMF_TIMEOUT	6	

ClockUpdateMode

NAME	VALUE	DESCRIPTION
COUNTUP	0	
COUNTDOWN	1	
PAUSE	2	
RESUME	3	
CLEAR	4	

DeviceLevelStatus

NAME	VALUE	DESCRIPTION
ZERO_LEVEL_BARS	0	
ONE_LEVEL_BARS	1	
TWO_LEVEL_BARS	2	
THREE_LEVEL_BARS	3	
FOUR_LEVEL_BARS	4	
NOT_PROVIDED	5	

SystemAction

NAME	VALUE	DESCRIPTION
DEFAULT_ACTION	0	
STEAL_FOCUS	1	
KEEP_CONTEXT	2	

ButtonName

NAME	VALUE	DESCRIPTION
OK	0	
PLAY_PAUSE	1	
SEEKLEFT	2	
SEEKRIGHT	3	
TUNEUP	4	
TUNEDOWN	5	
PRESET_0	6	
PRESET_1	7	
PRESET_2	8	
PRESET_3	9	
PRESET_4	10	
PRESET_5	11	
PRESET_6	12	
PRESET_7	13	
PRESET_8	14	
PRESET_9	15	
CUSTOM_BUTTON	16	
SEARCH	17	

NAME	VALUE	DESCRIPTION
AC_MAX	18	CLIMATE Module
AC	19	CLIMATE Module
RECIRCULATE	20	CLIMATE Module
FAN_UP	21	CLIMATE Module
FAN_DOWN	22	CLIMATE Module
TEMP_UP	23	CLIMATE Module
TEMP_DOWN	24	CLIMATE Module
DEFROST_MAX	25	CLIMATE Module
DEFROST	26	CLIMATE Module
DEFROST_REAR	27	CLIMATE Module
UPPER_VENT	28	CLIMATE Module
LOWER_VENT	29	CLIMATE Module
VOLUME_UP	30	RADIO Module
VOLUME_DOWN	31	RADIO Module
EJECT	32	RADIO Module
SOURCE	33	RADIO Module
SHUFFLE	34	RADIO Module
REPEAT	35	RADIO Module

KeypressMode

NAME	VALUE	DESCRIPTION
SINGLE_KEYPRESS	0	
QUEUE_KEYPRESSES	1	
RESEND_CURRENT_ENTRY	2	

WiperStatus

NAME	VALUE	DESCRIPTION
OFF	0	
AUTO_OFF	1	
OFF_MOVING	2	
MAN_INT_OFF	3	
MAN_INT_ON	4	
MAN_LOW	5	
MAN_HIGH	6	
MAN_FLICK	7	
WASH	8	
AUTO_LOW	9	
AUTO_HIGH	10	
COURTESYWIPE	11	
AUTO_ADJUST	12	
STALLED	13	
NO_DATA_EXISTS	14	

LayoutMode

NAME	VALUE	DESCRIPTION
ICON_ONLY	0	
ICON_WITH_SEARCH	1	
LIST_ONLY	2	
LIST_WITH_SEARCH	3	
KEYBOARD	4	

Language

NAME	VALUE	DESCRIPTION
EN-US	0	English - US
ES-MX	1	Spanish - Mexico
FR-CA	2	French - Canada
DE-DE	3	German - Germany
ES-ES	4	Spanish - Spain
EN-GB	5	English - GB
RU-RU	6	Russian - Russia
TR-TR	7	Turkish - Turkey
PL-PL	8	Polish - Poland
FR-FR	9	French - France
IT-IT	10	Italian - Italy
SV-SE	11	Swedish - Sweden
PT-PT	12	Portuguese - Portugal
NL-NL	13	Dutch (Standard) - Netherlands
EN-AU	14	English - Australia
ZH-CN	15	Mandarin - China
ZH-TW	16	Mandarin - Taiwan
JA-JP	17	Japanese - Japan

NAME	VALUE	DESCRIPTION
AR-SA	18	Arabic - Saudi Arabia
KO-KR	19	Korean - South Korea
PT-BR	20	Portuguese - Brazil
CS-CZ	21	Czech - Czech Republic
DA-DK	22	Danish - Denmark
NO-NO	23	Norwegian - Norway
NL-BE	24	Dutch (Flemish) - Belgium
EL-GR	25	Greek - Greece
HU-HU	26	Hungarian - Hungary
FI-FI	27	Finnish - Finland
SK-SK	28	Slovak - Slovakia
EN-IN	29	English - India
TH-TH	30	Thai - Thailand
EN-SA	31	English - Middle East
HE-IL	32	Hebrew - Israel
RO-RO	33	Romanian - Romania
UK-UA	34	Ukrainian - Ukraine
ID-ID	35	Indonesian - Indonesia

NAME	VALUE	DESCRIPTION
VI-VN	36	Vietnamese - Vietnam
MS-MY	37	Malay - Malaysia
HI-IN	38	Hindi - India

FileType

NAME	VALUE	DESCRIPTION
GRAPHIC_BMP	0	
GRAPHIC_JPEG	1	
GRAPHIC_PNG	2	
AUDIO_WAVE	3	
AUDIO_MP3	4	
AUDIO_AAC	5	
BINARY	6	
JSON	7	

AudioType

NAME	VALUE	DESCRIPTION
PCM	0	

TBTState

NAME	VALUE	DESCRIPTION
ROUTE_UPDATE_REQUEST	0	
ROUTE_ACCEPTED	1	
ROUTE_REFUSED	2	
ROUTE_CANCELLED	3	
ETA_REQUEST	4	
NEXT_TURN_REQUEST	5	
ROUTE_STATUS_REQUEST	6	
ROUTE_SUMMARY_REQUEST	7	
TRIP_STATUS_REQUEST	8	
ROUTE_UPDATE_REQUEST_TIMEOUT	9	

VehicleDataResultCode

NAME	VALUE	DESCRIPTION
SUCCESS	0	
TRUNCATED_DATA	1	
DISALLOWED	2	
USER_DISALLOWED	3	
INVALID_ID	4	
VEHICLE_DATA_NOT_AVAILABLE	5	
DATA_ALREADY_SUBSCRIBED	6	
DATA_NOT_SUBSCRIBED	7	
IGNORED	8	

DisplayType

NAME	VALUE	DESCRIPTION
CID	0	
TYPE2	1	
TYPE5	2	
NGN	3	
GEN2_8_DMA	4	
GEN2_6_DMA	5	
MFD3	6	
MFD4	7	
MFD5	8	
GEN3_8_INCH	9	

ApplicationExitReason

NAME	VALUE	DESCRIPTION
DRIVER_DISTRACTION_VIOLATION	0	By getting this value, SDL puts the named app to NONE HMILevel
USER_EXIT	1	By getting this value, SDL puts the named app to NONE HMILevel
UNAUTHORIZED_TRANSPORT_REGISTRATION	2	By getting this value, SDL unregisters the named application
UNSUPPORTED_HMI_RESOURCE	3	By getting this value, SDL unregisters the named application
CLOSE_CLOUD_CONNECTION	4	By getting this value, SDL puts the named app to NONE HMILevel. Used by the HMI to close a cloud app connection

IgnitionStatus

NAME	VALUE	DESCRIPTION
UNKNOWN	0	
OFF	1	
ACCESSORY	2	
RUN	3	
START	4	
INVALID	5	

EmergencyEventType

NAME	VALUE	DESCRIPTION
NO_EVENT	0	
FRONTAL	1	
SIDE	2	
REAR	3	
ROLLOVER	4	
NOT_SUPPORTED	5	
FAULT	6	

HMILevel

NAME	VALUE	DESCRIPTION
FULL	0	
LIMITED	1	
BACKGROUND	2	
NONE	3	

AppPriority

NAME	VALUE	DESCRIPTION
EMERGENCY	0	
NAVIGATION	1	
VOICE_COMMUNICATION	2	
COMMUNICATION	3	
NORMAL	4	
NONE	5	

PowerModeQualificationStatus

NAME	VALUE	DESCRIPTION
POWER_MODE_UNDEFINED	0	
POWER_MODE_EVALUATION_IN_PROGRESS	1	
NOT_DEFINED	2	
POWER_MODE_OK	3	

ApplicationsCloseReason

NAME	VALUE	DESCRIPTION
IGNITION_OFF	0	
MASTER_RESET	1	
FACTORY_DEFAULTS	2	
SUSPEND	3	

Dimension

NAME	VALUE	DESCRIPTION
NO_FIX	0	
2D	1	
3D	2	

ImageType

NAME	VALUE	DESCRIPTION
STATIC	0	
DYNAMIC	1	

AudioStreamingIndicator

NAME	VALUE	DESCRIPTION
PLAY_PAUSE	0	Default playback indicator.
PLAY	1	Indicates that a button press of the Play/Pause button would start the playback.
PAUSE	2	Indicates that a button press of the Play/Pause button would pause the current playback.
STOP	3	Indicates that a button press of the Play/Pause button would stop the current playback.

IgnitionStableStatus

NAME	VALUE	DESCRIPTION
IGNITION_SWITCH_NOT_STABLE	0	
IGNITION_SWITCH_STABLE	1	
MISSING_FROM_TRANSMITTER	2	

WarningLightStatus

NAME	VALUE	DESCRIPTION
OFF	0	
ON	1	
FLASH	2	
NOT_USED	3	

VehicleDataNotificationStatus

NAME	VALUE	DESCRIPTION
NOT_SUPPORTED	0	
NORMAL	1	
ACTIVE	2	
NOT_USED	3	

SystemError

NAME	VALUE	DESCRIPTION
SYNC_REBOOTED	0	
SYNC_OUT_OF_MEMMORY	1	

PowerModeStatus

NAME	VALUE	DESCRIPTION
KEY_OUT	0	
KEY_RECENTLY_OUT	1	
KEY_APPROVED_0	2	
POST_ACCESORY_0	3	
ACCESORY_1	4	
POST_IGNITION_1	5	
IGNITION_ON_2	6	
RUNNING_2	7	
CRANK_3	8	

ComponentVolumeStatus

NAME	VALUE	DESCRIPTION
UNKNOWN	0	
NORMAL	1	
LOW	2	
FAULT	3	
ALERT	4	
NOT_SUPPORTED	5	

PrimaryAudioSource

NAME	VALUE	DESCRIPTION
NO_SOURCE_SELECTED	0	
CD	1	
USB	2	
USB2	3	
BLUETOOTH_STEREO_BTST	4	
LINE_IN	5	
IPOD	6	
MOBILE_APP	7	
AM	8	
FM	9	
XM	10	
DAB	11	

RequestType

NAME	VALUE	DESCRIPTION
HTTP	0	
FILE_RESUME	1	
AUTH_REQUEST	2	
AUTH_CHALLENGE	3	
AUTH_ACK	4	
PROPRIETARY	5	
QUERY_APPS	6	
LAUNCH_APP	7	
LOCK_SCREEN_ICON_URL	8	
TRAFFIC_MESSAGE_CHANNEL	9	
DRIVER_PROFILE	10	
VOICE_SEARCH	11	
NAVIGATION	12	
PHONE	13	
CLIMATE	14	
SETTINGS	15	
VEHICLE_DIAGNOSTICS	16	
EMERGENCY	17	

NAME	VALUE	DESCRIPTION
MEDIA	18	
FOTA	19	
OEM_SPECIFIC	20	
ICON_URL	21	

ConsentSource

NAME	VALUE	DESCRIPTION
GUI	0	
VUI	1	

CompassDirection

NAME	VALUE	DESCRIPTION
NORTH	0	
NORTHWEST	1	
WEST	2	
SOUTHWEST	3	
SOUTH	4	
SOUTHEAST	5	
EAST	6	
NORTHEAST	7	

BitsPerSample

NAME	VALUE	DESCRIPTION
8_BIT	0	
16_BIT	1	

EmergencyState

NAME	VALUE	DESCRIPTION
EMERGENCY_ON	0	
EMERGENCY_OFF	1	

Result

NAME	VALUE	DESCRIPTION
SUCCESS	0	
UNSUPPORTED_REQUEST	1	
UNSUPPORTED_RESOURCE	2	If this response is returned to core and the related HMI interface state is not available, core will return `success:false` to mobile. Otherwise core will return `success:true`
DISALLOWED	3	
REJECTED	4	
ABORTED	5	
IGNORED	6	
RETRY	7	
IN_USE	8	
DATA_NOT_AVAILABLE	9	
TIMED_OUT	10	
INVALID_DATA	11	
CHAR_LIMIT_EXCEEDED	12	
INVALID_ID	13	
DUPLICATE_NAME	14	

NAME	VALUE	DESCRIPTION
APPLICATION_NOT_REGISTERED	15	
WRONG_LANGUAGE	16	
OUT_OF_MEMORY	17	
TOO_MANY_PENDING_REQUESTS	18	
NO_APPS_REGISTERED	19	
NO_DEVICES_CONNECTED	20	
WARNINGS	21	
GENERIC_ERROR	22	
USER_DISALLOWED	23	
TRUNCATED_DATA	24	

CarModeStatus

NAME	VALUE	DESCRIPTION
NORMAL	0	
FACTORY	1	
TRANSPORT	2	
CRASH	3	

ImageFieldName

NAME	VALUE	DESCRIPTION
softButtonImage	0	
choiceImage	1	
choiceSecondaryImage	2	
vrHelpItem	3	
turnIcon	4	
menuIcon	5	
cmdIcon	6	
appIcon	7	
graphic	8	
secondaryGraphic	9	
showConstantTBTIcon	10	
showConstantTBTNextTurnIcon	11	
locationImage	12	

VehicleDataType

NAME	VALUE	DESCRIPTION
VEHICLEDATA_GPS	0	
VEHICLEDATA_SPEED	1	
VEHICLEDATA_RPM	2	
VEHICLEDATA_FUELLEVEL	3	
VEHICLEDATA_FUELLEVEL_STATE	4	
VEHICLEDATA_FUELCONSUMPTION	5	
VEHICLEDATA_EXTERNTEMP	6	
VEHICLEDATA_VIN	7	
VEHICLEDATA_PRNDL	8	
VEHICLEDATA_TIREPRESSURE	9	
VEHICLEDATA_ODOMETER	10	
VEHICLEDATA_BELTSTATUS	11	
VEHICLEDATA_BODYINFO	12	
VEHICLEDATA_DEVICESTATUS	13	
VEHICLEDATA_ECALLINFO	14	
VEHICLEDATA_AIRBAGSTATUS	15	
VEHICLEDATA_EMERGENCYEVENT	16	
VEHICLEDATA_CLUSTERMODESTATUS	17	

NAME	VALUE	DESCRIPTION
VEHICLEDATA_MYKEY	18	
VEHICLEDATA_BRAKING	19	
VEHICLEDATA_WIPERSTATUS	20	
VEHICLEDATA_HEADLAMPSTATUS	21	
VEHICLEDATA_BATTVOLTAGE	22	
VEHICLEDATA_ENGINETORQUE	23	
VEHICLEDATA_ACCPEDAL	24	
VEHICLEDATA_STEERINGWHEEL	25	
VEHICLEDATA_TURNSIGNAL	26	
VEHICLEDATA_FUELRANGE	27	
VEHICLEDATA_ENGINEOILLIFE	28	
VEHICLEDATA_ELECTRONICPARKBRAKESTATUS	29	
VEHICLEDATA_CLOUDAPPVEHICLEID	30	Parameter used by cloud apps or the policy server to identify a head unit

VideoStreamingProtocol

NAME	VALUE	DESCRIPTION
RAW	0	Raw stream bytes
RTP	1	Real-time Transport Protocol
RTSP	2	Real-time Streaming Protocol
RTMP	3	Real-Time Messaging Protocol
WEBM	4	WebM container

VideoStreamingCodec

NAME	VALUE	DESCRIPTION
H264	0	MPEG-4 Advanced Video Coding
H265	1	High Efficiency Video Coding
Theora	2	Ogg Theora
VP8	3	
VP9	4	

UpdateResult

NAME	VALUE	DESCRIPTION
UP_TO_DATE	0	
UPDATING	1	
UPDATE_NEEDED	2	

PRNDL

NAME	VALUE	DESCRIPTION
PARK	0	
REVERSE	1	
NEUTRAL	2	
DRIVE	3	
SPORT	4	
LOWGEAR	5	
FIRST	6	
SECOND	7	
THIRD	8	
FOURTH	9	
FIFTH	10	
SIXTH	11	
SEVENTH	12	
EIGHTH	13	
UNKNOWN	14	
FAULT	15	

TPMS

NAME	VALUE	DESCRIPTION
UNKNOWN	0	If set the status of the tire is not known.
SYSTEM_FAULT	1	TPMS does not function.
SENSOR_FAULT	2	The sensor of the tire does not function.
LOW	3	TPMS is reporting a low tire pressure for the tire.
SYSTEM_ACTIVE	4	TPMS is active and the tire pressure is monitored.
TRAIN	5	TPMS is reporting that the tire must be trained.
TRAINING_COMPLETE	6	TPMS reports the training for the tire is completed.
NOT_TRAINED	7	TPMS reports the tire is not trained.

VehicleDataStatus

NAME	VALUE	DESCRIPTION
NO_DATA_EXISTS	0	
OFF	1	
ON	2	

ModuleType

NAME	VALUE	DESCRIPTION
CLIMATE	0	
RADIO	1	
SEAT	2	
AUDIO	3	
LIGHT	4	
HMI_SETTINGS	5	

RadioBand

NAME	VALUE	DESCRIPTION
AM	0	
FM	1	
XM	2	

RadioState

NAME	VALUE	DESCRIPTION
ACQUIRING	0	
ACQUIRED	1	
MULTICAST	2	
NOT_FOUND	3	

TemperatureUnit

NAME	VALUE	DESCRIPTION
FAHRENHEIT	0	
CELSIUS	1	

DefrostZone

NAME	VALUE	DESCRIPTION
FRONT	0	
REAR	1	
ALL	2	
NONE	3	

VentilationMode

NAME	VALUE	DESCRIPTION
UPPER	0	
LOWER	1	
BOTH	2	
NONE	3	

RCAccessMode

NAME	VALUE	DESCRIPTION
AUTO_ALLOW	0	Any RC app can take immediately gain control a module when it is requested
AUTO_DENY	1	An RC app has control of a module until the app releases it, all requests for control of the module by other apps are rejected
ASK_DRIVER	2	The driver is prompted when an app requests control of a module that is currently being used

EntityStatus

NAME	VALUE	DESCRIPTION
ON	0	
OFF	1	

FuelType

NAME	VALUE	DESCRIPTION
GASOLINE	0	
DIESEL	1	
CNG	2	For vehicles using compressed natural gas
LPG	3	For vehicles using liquefied petroleum gas
HYDROGEN	4	For FCEV (fuel cell electric vehicle)
BATTERY	5	For BEV (Battery Electric Vehicle), PHEV (Plug-in Hybrid Electric Vehicle), solar vehicles and other vehicles which run on a battery

TurnSignal

NAME	VALUE	DESCRIPTION
OFF	0	
LEFT	1	
RIGHT	2	
BOTH	3	

ElectronicParkBrakeStatus

NAME	VALUE	DESCRIPTION
CLOSED	0	Park brake actuators have been fully applied.
TRANSITION	1	Park brake actuators are transitioning to either Apply/Closed or Release/ Open state.
OPEN	2	Park brake actuators are released.
DRIVE_ACTIVE	3	When driver pulls the Electronic Park Brake switch while driving "at speed".
FAULT	4	When system has a fault or is under maintenance.

LightName

NAME	VALUE	DESCRIPTION
FRONT_LEFT_HIGH_BEAM	0	
FRONT_RIGHT_HIGH_BEAM	1	
FRONT_LEFT_LOW_BEAM	2	
FRONT_RIGHT_LOW_BEAM	3	
FRONT_LEFT_PARKING_LIGHT	4	
FRONT_RIGHT_PARKING_LIGHT	5	
FRONT_LEFT_FOG_LIGHT	6	
FRONT_RIGHT_FOG_LIGHT	7	
FRONT_LEFT_DAYTIME_RUNNING_LIGHT	8	
FRONT_RIGHT_DAYTIME_RUNNING_LIGHT	9	
FRONT_LEFT_TURN_LIGHT	10	
FRONT_RIGHT_TURN_LIGHT	11	
REAR_LEFT_FOG_LIGHT	12	
REAR_RIGHT_FOG_LIGHT	13	
REAR_LEFT_TAIL_LIGHT	14	
REAR_RIGHT_TAIL_LIGHT	15	
REAR_LEFT_BRAKE_LIGHT	16	
REAR_RIGHT_BRAKE_LIGHT	17	

NAME	VALUE	DESCRIPTION
REAR_LEFT_TURN_LIGHT	18	
REAR_RIGHT_TURN_LIGHT	19	
REAR_REGISTRATION_PLATE_LIGHT	20	
HIGH_BEAMS	501	
LOW_BEAMS	502	
FOG_LIGHTS	503	
RUNNING_LIGHTS	504	
PARKING_LIGHTS	505	
BRAKE_LIGHTS	506	
REAR_REVERSING_LIGHTS	507	
SIDE_MARKER_LIGHTS	508	
LEFT_TURN_LIGHTS	509	
RIGHT_TURN_LIGHTS	510	
HAZARD_LIGHTS	511	
REAR_CARGO_LIGHTS	512	Cargo lamps illuminate the cargo area.
REAR_TRUCK_BED_LIGHTS	513	Truck bed lamps light up the bed of the truck.
REAR_TRAILER_LIGHTS	514	Trailer lights are lamps mounted on a trailer hitch.

NAME	VALUE	DESCRIPTION
LEFT_SPOT_LIGHTS	515	It is the spotlights mounted on the left side of a vehicle.
RIGHT_SPOT_LIGHTS	516	It is the spotlights mounted on the right side of a vehicle.
LEFT_PUDDLE_LIGHTS	517	Puddle lamps illuminate the ground beside the door as the customer is opening or approaching the door.
RIGHT_PUDDLE_LIGHTS	518	Puddle lamps illuminate the ground beside the door as the customer is opening or approaching the door.
AMBIENT_LIGHTS	801	
OVERHEAD_LIGHTS	802	
READING_LIGHTS	803	
TRUNK_LIGHTS	804	

LightStatus

NAME	VALUE	DESCRIPTION
ON	0	
OFF	1	
RAMP_UP	2	
RAMP_DOWN	3	
UNKNOWN	4	
INVALID	5	

DisplayMode

NAME	VALUE	DESCRIPTION
DAY	0	
NIGHT	1	
AUTO	2	

DistanceUnit

NAME	VALUE	DESCRIPTION
MILES	0	
KILOMETERS	1	

MassageZone

NAME	VALUE	DESCRIPTION
OFF	0	
LOW	1	
HIGH	2	

MassageCushion

NAME	VALUE	DESCRIPTION
TOP_LUMBAR	0	
MIDDLE_LUMBAR	1	
BOTTOM_LUMBAR	2	
BACK_BOLSTERS	3	
SEAT_BOLSTERS	4	

SeatMemoryActionType

NAME	VALUE	DESCRIPTION
SAVE	0	Save current seat postions and settings to seat memory.
RESTORE	1	Restore / apply the seat memory settings to the current seat.
NONE	2	No action to be performed.

SupportedSeat

NAME	VALUE	DESCRIPTION
DRIVER	0	List possible seats that is a remote controllable seat.
FRONT_PASSENGER	1	List possible seats that is a remote controllable seat.

DeliveryMode

NAME	VALUE	DESCRIPTION
PROMPT	0	
DESTINATION	1	
QUEUE	2	

AppServiceType

NAME	VALUE	DESCRIPTION
MEDIA	0	
WEATHER	1	
NAVIGATION	2	

ServiceUpdateReason

NAME	VALUE	DESCRIPTION
PUBLISHED	0	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	1	The service has just been unpublished with the module and is no longer accessible
ACTIVATED	2	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	3	The service has been deactivated as the primary service of its type
MANIFEST_UPDATE	4	The service has updated its manifest. This could imply updated capabilities. **Note: Currently unimplemented**

SystemCapabilityType

NAME	VALUE	DESCRIPTION
NAVIGATION	0	
PHONE_CALL	1	
VIDEO_STREAMING	2	
REMOTE_CONTROL	3	
APP_SERVICES	4	

MediaType

NAME	VALUE	DESCRIPTION
MUSIC	0	
PODCAST	1	
AUDIOBOOK	2	
OTHER	3	

NavigationAction

NAME	VALUE	DESCRIPTION
TURN	0	Using this action plus a supplied direction can give the type of turn
EXIT	1	
STAY	2	
MERGE	3	
FERRY	4	
CAR_SHUTTLE_TRAIN	5	
WAYPOINT	6	

NavigationJunction

NAME	VALUE	DESCRIPTION
REGULAR	0	A junction that represents a standard intersection with a single road crossing another
BIFURCATION	1	A junction where the road splits off into two paths; a fork in the road
MULTI_CARRIAGEWAY	2	A junction that has multiple intersections and paths
ROUNDABOUT	3	A junction where traffic moves in a single direction around a central, non-traversable point to reach one of the connecting roads
TRAVERSABLE_ROUNDABOUT	4	Similar to a roundabout, however the center of the roundabout is fully traversable. Also known as a mini-roundabout
JUGHANDLE	5	A junction where lefts diverge to the right, then curve to the left, converting a left turn to a crossing maneuver
ALL_WAY_YIELD	6	Multiple way intersection that allows traffic to flow based on priority; most commonly right of way and first in, first out
TURN_AROUND	7	A junction designated for traffic turn arounds

Direction

NAME	VALUE	DESCRIPTION
LEFT	0	
RIGHT	1	

Structs

ImageResolution

NAME	TYPE	MANDATORY	ADDITIONAL	DESCRIPTION
resolutionWidth	Integer	true	minvalue: 1 maxvalue: 10000	
resolutionHeight	Integer	true	minvalue: 1 maxvalue: 10000	

UserFriendlyMessage

NAME	TYPE	MANDATORY	ADDITIONAL	DESCRIPTION
messageCode	String	true		
ttsString	String	false		
label	String	false		
line1	String	false		
line2	String	false		
textBody	String	false		

HMICapabilities

NAME	TYPE	MANDATORY	ADDITIONAL	DESCRIPTION
navigation	Boolean	false		
phoneCall	Boolean	false		

NavigationCapability

NAME	TYPE	MANDATORY	ADDITIONAL	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

NAME	TYPE	MANDATORY	ADDITIONAL	DESCRIPTION
dialNumberEnabled	Boolean	false		If the module has the ability to perform dial number

VideoStreamingCapability

NAME	TYPE	MANDATORY	ADDITIONAL	DESCRIPTION
preferredResolution	Common.ImageResolution	false		The preferred resolution of a video stream for decoding and rendering on HMI.
maxBitrate	Integer	false	minvalue: 0 maxvalue: 2147483647	The maximum bitrate of video stream that is supported, in kbps.
supportedFormats	Common.VideoStreamingFormat	false	array: true	Detailed information on each format supported by this system, in its preferred order.
hapticSpatialDataSupported	boolean	false		True if the system can utilize the haptic spatial data from the source being streamed.

SystemCapabilities

NAME	TYPE	MANDATORY	ADDITIONAL	DESCRIPTION
navigationCapability	Common.NavigationCapability	false		
phoneCapability	Common.PhoneCapability	false		
videoStreamingCapability	Common.VideoStreamingCapability	false		

MenuParams

NAME	TYPE	MANDATORY	ADDITIONAL	DESCRIPTION
parentID	Integer	false	minvalue: 0 maxvalue: 2000000000	
position	Integer	false	minvalue: 0 maxvalue: 1000	
menuName	String	true	maxlength: 500	

TireStatus

NAME	TYPE	MANDATORY	ADDITIONAL	DESCRIPTION
pressureTelltale	Common.WarningLightStatus	false		
leftFront	Common.SingleTireStatus	false		
rightFront	Common.SingleTireStatus	false		
leftRear	Common.SingleTireStatus	false		
rightRear	Common.SingleTireStatus	false		
innerLeftRear	Common.SingleTireStatus	false		
innerRightRear	Common.SingleTireStatus	false		

ECallInfo

NAME	TYPE	MANDATORY	ADDITIONAL	DESCRIPTION
eCallNotificationStatus	Common.VehicleDataNotificationStatus			
auxECallNotificationStatus	Common.VehicleDataNotificationStatus			
eCallConfirmationStatus	Common.ECallConfirmationStatus			

DIDResult

NAME	TYPE	MANDATORY	ADDITIONAL	DESCRIPTION
resultCode	Common.VehicleDataResultCode	true		
didLocation	Integer	true	minvalue: 0 maxvalue: 65535	
data	String	false	maxlength: 5000	

TTSChunk

NAME	TYPE	MANDATORY	ADDITIONAL	DESCRIPTION
text	String	true	maxlength: 500	The text/ phonemes to be spoken or the name of an audio file to play
type	Common.SpeechCapabilities	true		Describes how to interpret the text field (as plain text, a file name, etc.)

TextField

NAME	TYPE	MANDATORY	ADDITIONAL	DESCRIPTION
name	Common.TextFieldName			
characterSet	Common.CharacterSet			
width	Integer		minvalue: 1 maxvalue: 500	
rows	Integer		minvalue: 1 maxvalue: 8	

TouchCoord

NAME	TYPE	MANDATORY	ADDITIONAL	DESCRIPTION
x	Integer	true	minvalue: 0 maxvalue: 10000	
y	Integer	true	minvalue: 0 maxvalue: 10000	

AudioPassThruCapabilities

NAME	TYPE	MANDATORY	ADDITIONAL	DESCRIPTION
samplingRate	Common.SamplingRate	true		
bitsPerSample	Common.BitsPerSample	true		
audioType	Common.AudioType	true		

ServiceInfo

NAME	TYPE	MANDATORY	ADDITIONAL	DESCRIPTION
url	String	true		
policyAppId	String	false		

HeadLampStatus

NAME	TYPE	MANDATORY	ADDITIONAL	DESCRIPTION
lowBeamsOn	Boolean	true		
highBeamsOn	Boolean	true		
ambientLightSensorStatus	Common.AmbientLightStatus	true		

ClusterModeStatus

NAME	TYPE	MANDATORY	ADDITIONAL	DESCRIPTION
powerModeActive	Boolean			
powerModeQualificationStatus	Common.PowerModeQualificationStatus			
carModeStatus	Common.CarModeStatus			
powerModeStatus	Common.PowerModeStatus			

KeyboardProperties

NAME	TYPE	MANDATORY	ADDITIONAL	DESCRIPTION
language	Common.Language	false		
keyboardLayout	Common.KeyboardLayout	false		
keypressMode	Common.KeypressMode	false		
limitedCharacterList	String	false	array: true minsize: 1 maxsize: 100 maxlength: 1	
autoCompleteText	String	false	maxlength: 1000	

Choice

NAME	TYPE	MANDATORY	ADDITIONAL	DESCRIPTION
choiceID	Integer	true	minvalue: 0 maxvalue: 65535	
menuName	String	false	maxlength: 500	
image	Common.Image	false		
secondaryText	String	false	maxlength: 500	
tertiaryText	String	false	maxlength: 500	
secondaryImage	Image	false		

DeviceStatus

NAME	TYPE	MANDATORY	ADDITIONAL	DESCRIPTION
voiceRecOn	Boolean	false		
btIconOn	Boolean	false		
callActive	Boolean	false		
phoneRoaming	Boolean	false		
textMsgAvailable	Boolean	false		
battLevelStatus	Common.DeviceLevelStatus	false		
stereoAudioOutputMuted	Boolean	false		
monoAudioOutputMuted	Boolean	false		
signalLevelStatus	Common.DeviceLevelStatus	false		
primaryAudioSource	Common.PrimaryAudioSource	false		
eCallEventActive	Boolean	false		

GPSData

NAME	TYPE	MANDATORY	ADDITIONAL	DESCRIPTION
longitudeDegrees	Float	true	minvalue: -180 maxvalue: 180	
latitudeDegrees	Float	true	minvalue: -90 maxvalue: 90	
utcYear	Integer	false	minvalue: 2010 maxvalue: 2100	
utcMonth	Integer	false	minvalue: 1 maxvalue: 12	
utcDay	Integer	false	minvalue: 1 maxvalue: 31	
utcHours	Integer	false	minvalue: 0 maxvalue: 23	
utcMinutes	Integer	false	minvalue: 0 maxvalue: 59	
utcSeconds	Integer	false	minvalue: 0 maxvalue: 59	
compassDirection	Common.CompassDirection	false		
pdop	Float	false	minvalue: 0 maxvalue: 1000	
hdop	Float	false	minvalue: 0 maxvalue: 1000	
vdop	Float	false	minvalue: 0 maxvalue: 1000	

NAME	TYPE	MANDATORY	ADDITIONAL	DESCRIPTION
actual	Boolean	false		
satellites	Integer	false	minvalue: 0 maxvalue: 31	
dimension	Common.Dimension	false		
altitude	Float	false	minvalue: -10000 maxvalue: 10000	
heading	Float	false	minvalue: 0 maxvalue: 359.99	
speed	Float	false	minvalue: 0 maxvalue: 500	

SingleTireStatus

NAME	TYPE	MANDATORY	ADDITIONAL	DESCRIPTION
status	Common.ComponentVolumeStatus	true		
tpms	Common.TPMS	false		The status of TPMS according to the particular tire.
pressure	Float	false	minvalue: 0 maxvalue: 2000	The pressure value of the particular tire in kilopascals.

SoftButtonCapabilities

NAME	TYPE	MANDATORY	ADDITIONAL	DESCRIPTION
shortPressAvailable	Boolean	true		
longPressAvailable	Boolean	true		
upDownAvailable	Boolean	true		
imageSupported	Boolean	true		

HMIApplication

NAME	TYPE	MANDATORY	ADDITIONAL	DESCRIPTION
appName	String	true	maxlength: 100	
ngnMediaScreenAppName	String	false	maxlength: 100	
icon	String	false		
deviceInfo	Common.DeviceInfo	true		
secondaryDeviceInfo	Common.DeviceInfo	true		The ID, serial number, transport type that are acquired through Secondary Transport.
policyAppID	String	true	minlength: 1 maxlength: 50	
ttsName	Common.TTSChunk	false	array: true minsize: 1 maxsize: 100	
vrSynonyms	String	false	array: true minsize: 1 maxsize: 100 maxlength: 40	
appID	Integer	true		
hmiDisplayLanguageDesired	Common.Language	false		
isMediaApplication	Boolean	false		

NAME	TYPE	MANDATORY	ADDITIONAL	DESCRIPTION
appType	Common.AppHMIType	false	array: true minsize: 1 maxsize: 100	
greyOut	Boolean	false		
requestType	Common.RequestType	false	array: true minsize: 0 maxsize: 100	

NAME	TYPE	MANDATORY	ADDITIONAL	DESCRIPTION
requestSubType	String	false	array:true minsize:0 maxsize:100 maxlength:100	The list of SystemRequest's requestSubTypes allowed by policies for the named application. If the app sends a requestSubType which is not specified in this list, then that request should be rejected. An empty array signifies that any value of requestSubType is allowed for this app. If this parameter is omitted, then a request with any value of requestSubType is now allowed for this app

NAME	TYPE	MANDATORY	ADDITIONAL	DESCRIPTION
dayColorScheme	Common.TemplateColorScheme	false		
nightColorScheme	Common.TemplateColorScheme	false		
isCloudApplication	Boolean	false		
cloudConnectionStatus	Common.CloudConnectionStatus	false		

VehicleType

NAME	TYPE	MANDATORY	ADDITIONAL	DESCRIPTION
make	String	false	maxlength: 500	
model	String	false	maxlength: 500	
modelYear	String	false	maxlength: 500	
trim	String	false	maxlength: 500	

ButtonCapabilities

NAME	TYPE	MANDATORY	ADDITIONAL	DESCRIPTION
name	Common.ButtonName	true		
shortPressAvailable	Boolean	true		
longPressAvailable	Boolean	true		
upDownAvailable	Boolean	true		

VrHelpItem

NAME	TYPE	MANDATORY	ADDITIONAL	DESCRIPTION
text	String	true	maxlength: 500	
image	Common.Image	false		
position	Integer	true	minvalue: 1 maxvalue: 100	

BodyInformation

NAME	TYPE	MANDATORY	ADDITIONAL	DESCRIPTION
parkBrakeActive	Boolean	true		
ignitionStableStatus	Common.IgnitionStableStatus	true		
ignitionStatus	Common.IgnitionStatus	true		
driverDoorAjar	Boolean	false		
passengerDoorAjar	Boolean	false		
rearLeftDoorAjar	Boolean	false		
rearRightDoorAjar	Boolean	false		

BeltStatus

NAME	TYPE	MANDATORY	ADDITIONAL	DESCRIPTION
driverBeltDeployed	Common.VehicleDataEventStatus	false		
passengerBeltDeployed	Common.VehicleDataEventStatus	false		
passengerBuckleBelted	Common.VehicleDataEventStatus	false		
driverBuckleBelted	Common.VehicleDataEventStatus	false		
leftRow2BuckleBelted	Common.VehicleDataEventStatus	false		
passengerChildDetected	Common.VehicleDataEventStatus	false		
rightRow2BuckleBelted	Common.VehicleDataEventStatus	false		
middleRow2BuckleBelted	Common.VehicleDataEventStatus	false		
middleRow3BuckleBelted	Common.VehicleDataEventStatus	false		
leftRow3BuckleBelted	Common.VehicleDataEventStatus	false		
rightRow3BuckleBelted	Common.VehicleDataEventStatus	false		
leftRearInflatableBelted	Common.VehicleDataEventStatus	false		
rightRearInflatableBelted	Common.VehicleDataEventStatus	false		

NAME	TYPE	MANDATORY	ADDITIONAL	DESCRIPTION
middleRow1BeltDeployed	Common.VehicleDataEventStatus	false		
middleRow1BuckleBelted	Common.VehicleDataEventStatus	false		

Turn

NAME	TYPE	MANDATORY	ADDITIONAL	DESCRIPTION
navigationText	Common.TextFieldStruct	false		
turnIcon	Common.Image	false		

EmergencyEvent

NAME	TYPE	MANDATORY	ADDITIONAL	DESCRIPTION
emergencyEventType	Common.EmergencyEventType			
fuelCutoffStatus	Common.FuelCutoffStatus			
rolloverEvent	Common.VehicleDataEventStatus			
maximumChangeVelocity	Common.VehicleDataEventStatus			
multipleEvents	Common.VehicleDataEventStatus			

VehicleDataResult

NAME	TYPE	MANDATORY	ADDITIONAL	DESCRIPTION
dataType	Common.VehicleDataType			
resultCode	Common.VehicleDataResultCode			

PresetBankCapabilities

NAME	TYPE	MANDATORY	ADDITIONAL	DESCRIPTION
onScreenPresetsAvailable	Boolean	true		

TouchEvent

NAME	TYPE	MANDATORY	ADDITIONAL	DESCRIPTION
id	Integer	true	minvalue: 0 maxvalue: 9	
ts	Integer	true	array: true minsize: 1 maxsize: 1000 minvalue: 0 maxvalue: 2147483647	
c	Common.TouchCoord	true	array: true minsize: 1 maxsize: 1000	

PermissionItem

NAME	TYPE	MANDATORY	ADDITIONAL	DESCRIPTION
name	String	true		
id	Integer	true		
allowed	Boolean	false		

TouchEventCapabilities

NAME	TYPE	MANDATORY	ADDITIONAL	DESCRIPTION
pressAvailable	Boolean	true		
multiTouchAvailable	Boolean	true		
doublePressAvailable	Boolean	true		

ScreenParams

NAME	TYPE	MANDATORY	ADDITIONAL	DESCRIPTION
resolution	Common.ImageResolution	true		
touchEventAvailable	Common.TouchEventCapabilities	false		

TextFieldStruct

NAME	TYPE	MANDATORY	ADDITIONAL	DESCRIPTION
fieldName	Common.TextFieldName	true		The name of the field for displaying the text.
fieldText	String	true	maxlength: 500	The data contained in the field.
fieldType	Common.MetadataType	false	array: true minSize: 0 maxSize: 5	The type of data contained in the field.

DeviceInfo

NAME	TYPE	MANDATORY	ADDITIONAL	DESCRIPTION
name	String	true		
id	String	true		
transportType	Common.TransportType	false		
isSDLAllowed	Boolean	false		

SoftButton

NAME	TYPE	MANDATORY	ADDITIONAL	DESCRIPTION
type	Common.SoftButtonType	true		
text	String	false	maxlength: 500	
image	Common.Image	false		
isHighlighted	Boolean	false		
softButtonID	Integer	true	minvalue: 0 maxvalue: 65535	
systemAction	Common.SystemAction	true		

AirbagStatus

NAME	TYPE	MANDATORY	ADDITIONAL	DESCRIPTION
driverAirbagDeployed	Common.VehicleDataEventStatus			
driverSideAirbagDeployed	Common.VehicleDataEventStatus			
driverCurtainAirbagDeployed	Common.VehicleDataEventStatus			
passengerAirbagDeployed	Common.VehicleDataEventStatus			
passengerCurtainAirbagDeployed	Common.VehicleDataEventStatus			
driverKneeAirbagDeployed	Common.VehicleDataEventStatus			
passengerSideAirbagDeployed	Common.VehicleDataEventStatus			
passengerKneeAirbagDeployed	Common.VehicleDataEventStatus			

RGBColor

NAME	TYPE	MANDATORY	ADDITIONAL	DESCRIPTION
red	Integer	true	minvalue: 0 maxvalue: 255	
green	Integer	true	minvalue: 0 maxvalue: 255	
blue	Integer	true	minvalue: 0 maxvalue: 255	

TemplateColorScheme

NAME	TYPE	MANDATORY	ADDITIONAL	DESCRIPTION
primaryColor	Common.RGBColor	true		
secondaryColor	Common.RGBColor	true		
backgroundColor	Common.RGBColor	true		

DisplayCapabilities

NAME	TYPE	MANDATORY	ADDITIONAL	DESCRIPTION
displayType	Common.DisplayType	true		
displayName	String	true		The name of the display the app is connected to.
textFields	Common.TextField	true	array: true minsize: 0 maxsize: 100	
imageFields	Common.ImageField	false	array: true minsize: 1 maxsize: 100	
mediaClockFormats	Common.MediaClockFormat	true	array: true minsize: 0 maxsize: 100	
imageCapabilities	Common.ImageType	false	array: true minsize: 0 maxsize: 2	
graphicSupported	Boolean	true		
templatesAvailable	String	true	array: true minsize: 0 maxsize: 100 maxlength: 100	
screenParams	Common.ScreenParams	false		
numCustomPresetsAvailable	Integer	false	minvalue: 1 maxvalue: 100	

TimeFormat

NAME	TYPE	MANDATORY	ADDITIONAL	DESCRIPTION
hours	Integer	true	minvalue: 0 maxvalue: 59	
minutes	Integer	true	minvalue: 0 maxvalue: 59	
seconds	Integer	true	minvalue: 0 maxvalue: 59	

Image

NAME	TYPE	MANDATORY	ADDITIONAL	DESCRIPTION
value	String	true	maxlength: 65535	The path to the dynamic image stored on HU or the static binary image itself. **Note:** There is no guarantee that the image reference is valid at the time it is received.
imageType	Common.ImageType	true		Describes whether it is a static or dynamic image.
isTemplate	Boolean	false		Optional value to specify whether it's a template image. A template image can be (re)colored by the HMI as needed by using an image pattern.

MyKey

NAME	TYPE	MANDATORY	ADDITIONAL	DESCRIPTION
e911Override	Common.VehicleDataStatus	true		

ImageField

NAME	TYPE	MANDATORY	ADDITIONAL	DESCRIPTION
name	Common.ImageFieldName	true		
imageTypeSupported	Common.FileType	false	array: true minsize: 1 maxsize: 100	
imageResolution	Common.ImageResolution	false		

VideoConfig

NAME	TYPE	MANDATORY	ADDITIONAL	DESCRIPTION
protocol	Common.VideoStreamingProtocol	false		The video protocol configuration.
codec	Common.VideoStreamingCodec	false		The video codec configuration.
width	Integer	false		Width of the video stream, in pixels.
height	Integer	false		Height of the video stream, in pixels.

ExternalConsentStatus

NAME	TYPE	MANDATORY	ADDITIONAL	DESCRIPTION
entityType	Integer	true	minvalue: 0 maxvalue: 128	The entityType which status is informed by "status" param.
entityID	Integer	true	minvalue: 0 maxvalue: 128	The corresponding ID of entityType which status is informed by "status" param.
status	Common.EntityStatus	true	-	Status of External User Consent Settings entity: "ON" or "OFF"

ModuleData

NAME	TYPE	MANDATORY	ADDITIONAL	DESCRIPTION
moduleType	Common.ModuleType	true		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
radioControlData	Common.RadioControlData	false		
climateControlData	Common.ClimateControlData	false		
audioControlData	Common.AudioControlData	false		
lightControlData	Common.LightControlData	false		
hmiSettingsControlData	Common.HMISettingsControlData	false		

RadioControlData

NAME	TYPE	MANDATORY	ADDITIONAL	DESCRIPTION
frequencyInteger	Integer	false	minvalue:0 maxvalue:1710	The integer part of the frequency ie for 101.7 this value should be 101
frequencyFraction	Integer	false	minvalue:0 maxvalue:9	The fractional part of the frequency for 101.7 is 7
band	Common.RadioBand	false		
rdsData	Common.RdsData	false		
availableHDs	Integer	false	minvalue:1 maxvalue:7	number of HD sub-channels if available
hdChannel	Integer	false	minvalue:1 maxvalue:7	Current HD sub-channel if available
signalStrength	Integer	false	minvalue:0 maxvalue:100	

NAME	TYPE	MANDATORY	ADDITIONAL	DESCRIPTION
signalChangeThreshold	Integer	false	minvalue:0 maxvalue:100	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 is the radio is off
state	Common.RadioState	false		

NAME	TYPE	MANDATORY	ADDITIONAL	DESCRIPTION
sisData	Common.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
hdRadioEnable	Boolean	false		True if the hd radio is on, false is the radio is off

RdsData

NAME	TYPE	MANDATORY	ADDITIONAL	DESCRIPTION
PS	String	false	minlength:0 maxlength:8	Program Service Name
RT	String	false	minlength:0 maxlength:64	Radio Text
CT	String	false	minlength:24 maxlength:24	The clock text in UTC format as YYYY-MM-DDThh:mm:ss.sTZD
PI	String	false	minlength:0 maxlength:6	Program Identification - the call sign for the radio station
PTY	Integer	false	minvalue:0 maxvalue:31	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

NAME	TYPE	MANDATORY	ADDITIONAL	DESCRIPTION
TA	Boolean	false		Traffic Announcement Identification - Indicates an ongoing traffic announcement
REG	String	false		Region

StationIDNumber

NAME	TYPE	MANDATORY	ADDITIONAL	DESCRIPTION
countryCode	Integer	false	minvalue="0" maxvalue="999"	Binary Representation of ITU Country Code. USA Code is 001.
fccFacilityId	Integer	false	minvalue="0" maxvalue="999999"	Binary representation of unique facility ID assigned by the FCC; FCC controlled for U.S. territory

SisData

NAME	TYPE	MANDATORY	ADDITIONAL	DESCRIPTION
stationShortName	String	false	minlength="4" maxlength="7"	Identifies the 4-alpha-character station call sign plus an optional (-FM) extension
stationIDNumber	Common.StationIDNumber	false		Used for network Application. Consists of Country Code and FCC Facility ID.
stationLongName	String	false	minlength="0" maxlength="56"	Identifies the station call sign or other identifying information in the long format.
stationLocation	Common.GPSData	false		Provides the 3-dimensional geographic station location.

NAME	TYPE	MANDATORY	ADDITIONAL	DESCRIPTION
stationMessage	String	false	minlength="0" maxlength="56"	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.

ClimateControlData

NAME	TYPE	MANDATORY	ADDITIONAL	DESCRIPTION
fanSpeed	Integer	false	minvalue: 0 maxvalue: 100	
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.
heatedWindshieldEnable	Boolean	false		value false means disabled, value true means enabled.

NAME	TYPE	MANDATORY	ADDITIONAL	DESCRIPTION
heatedRearWindowEnable	Boolean	false		value false means disabled, value true means enabled.
heatedMirrorsEnable	Boolean	false		value false means disabled, value true means enabled.

Temperature

NAME	TYPE	MANDATORY	ADDITIONAL	DESCRIPTION
unit	TemperatureUnit	true		TemperatureUnit
value	Float	true		The temperature value is in TemperatureUnit specified unit

RemoteControlCapabilities

NAME	TYPE	MANDATORY	ADDITIONAL	DESCRIPTION
climateControlCapabilities	ClimateControlCapabilities	false	array: true minsize: 1 maxsize: 100	If included, the platform supports RC climate controls. For this baseline version, maxsize= 1. i.e. only one climate control module is supported
radioControlCapabilities	RadioControl Capabilities	false	array: true minsize: 1 maxsize: 100	If included, the platform supports RC radio controls. For this baseline version, maxsize= 1. i.e. only one climate control module is supported
seatControl Capabilities	Common.SeatControlCapabilities	false	minsize="1" maxsize="100" array="true"	If included, the platform supports seat controls

NAME	TYPE	MANDATORY	ADDITIONAL	DESCRIPTION
buttonCapabilities	Common.ButtonCapabilities	false	array: true minsize: 1 maxsize: 100	If included, the platform supports RC button controls with the included button names
seatControlCapabilities	Common.SeatControlCapabilities	false	minsize="1" maxsize="100" array="true"	If included, the platform supports seat controls.
audioControlCapabilities	Common.AudioControlCapabilities	false	minsize="1" maxsize="100" array="true"	If included, the platform supports audio controls.
hmiSettingsControlCapabilities	Common.HMISettingsControlCapabilities	false		If included, the platform supports hmi setting controls.
lightControlCapabilities	Common.LightControlCapabilities	false		If included, the platform supports light controls.

ClimateControlCapabilities

NAME	TYPE	MANDATORY	ADDITIONAL	DESCRIPTION
moduleName	String	true	maxlength: 100	The short friendly name of the climate control module. It should not be used to identify a module by mobile application.
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.

NAME	TYPE	MANDATORY	ADDITIONAL	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.

NAME	TYPE	MANDATORY	ADDITIONAL	DESCRIPTION
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.

NAME	TYPE	MANDATORY	ADDITIONAL	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	array: true minsize: 1 maxsize: 100	A set of all defrost zones that are controllable.

NAME	TYPE	MANDATORY	ADDITIONAL	DESCRIPTION
ventilationModeAvailable	Boolean	false		Availability of the control of air ventilation mode. True: Available, False: Not Available, Not present: Not Available.
ventilationMode	VentilationMode	false	array: true minsize: 1 maxsize: 100	A set of all ventilation modes that are controllable
heatedSteeringWheelAvailable	Boolean	false		Availability of the control (enable/ disable) of heated Steering Wheel. True: Available, False: Not Available, Not present: Not Available.

NAME	TYPE	MANDATORY	ADDITIONAL	DESCRIPTION
heatedWindshieldAvailable	Boolean	false		Availability of the control (enable/ disable) of heated Windshield. True: Available, False: Not Available, Not present: Not Available.
heatedRearWindowAvailable	Boolean	false		Availability of the control (enable/ disable) of heated Rear Window. True: Available, False: Not Available, Not present: Not Available.

NAME	TYPE	MANDATORY	ADDITIONAL	DESCRIPTION
heatedMirrorsAvailable	Boolean	false		Availability of the control (enable/ disable) of heated Mirrors. True: Available, False: Not Available, Not present: Not Available.

AudioControlCapabilities

NAME	TYPE	MANDATORY	ADDITIONAL	DESCRIPTION
moduleName	String	true	maxlength="100"	The short friendly name of the light control module. It should not be used to identify a module by mobile application.
sourceAvailable	Boolean	false		Availability of the control of audio source.
keepContextAvailable	Boolean	false		Availability of the parameter keepContext.
volumeAvailable	Boolean	false		Availability of the control of audio volume.
equalizerAvailable	Boolean	false		Availability of the control of Equalizer Settings.

NAME	TYPE	MANDATORY	ADDITIONAL	DESCRIPTION
equalizerMaxChannelId	Integer	false	minvalue="1" maxvalue="100"	Must be included if equalizerAvailable=true, and assume all IDs starting from 1 to this value are valid.

EqualizerSettings

NAME	TYPE	MANDATORY	ADDITIONAL	DESCRIPTION
channelId	Integer	true	minvalue="1" maxvalue="100"	Defines the each Equalizer channel settings.
channelName	String	false	maxlength="50"	Read-only channel / frequency name (e.i. "Treble, Midrange, Bass" or "125 Hz")
channelSetting	Integer	true	minvalue="0" maxvalue="100"	Reflects the setting, from 0%-100%.

AudioControlData

NAME	TYPE	MANDATORY	ADDITIONAL	DESCRIPTION
source	Common.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.

			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 false, the system not only changes the audio source but also brings the default infotainment system UI associated with the audio source to foreground and set the application to background. If it is true, the system changes the audio source, but keeps the current application'
keepContext	Boolean	false	

NAME	TYPE	MANDATORY	ADDITIONAL	DESCRIPTION
volume	Integer	false	minvalue="0" maxvalue="100"	Reflects the volume of audio, from 0%-100%."
equalizerSettings	Common.EqualizerSettings	false	minvalue="1" maxvalue="100" array="true"	Defines the list of supported channels (band) and their current/ desired settings on HMI

LightCapabilities

NAME	TYPE	MANDATORY	ADDITIONAL	DESCRIPTION
name	Common.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

NAME	TYPE	MANDATORY	ADDITIONAL	DESCRIPTION
moduleName	String	true	maxlength="100"	The short friendly name of the light control module. It should not be used to identify a module by mobile application.
supportedLights	Common.LightCapabilities	true	minsize="1" maxsize="100" array="true"	An array of available LightCapabilities that are controllable.

LightState

NAME	TYPE	MANDATORY	ADDITIONAL	DESCRIPTION
id	Common.LightName	true		The name of a light or a group of lights.
status	Common.LightName	true		
density	Float	false	minvalue="0" maxvalue="1"	
color	Common.RGBColor	false		

LightControlData

NAME	TYPE	MANDATORY	ADDITIONAL	DESCRIPTION
lightState	Common.LightState	true	minsize="1" maxsize="100" array="true"	An array of LightNames and their current or desired status. Status of the LightNames that are not listed in the array shall remain unchanged.

HMISettingsControlCapabilities

NAME	TYPE	MANDATORY	ADDITIONAL	DESCRIPTION
moduleName	String	true	maxlength="100"	The short friendly name of the hmi setting module. It should not be used to identify a module by mobile application.
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.

HMISettingsControlData

NAME	TYPE	MANDATORY	ADDITIONAL	DESCRIPTION
displayMode	Common.DisplayMode	false		
temperatureUnit	Common.TemperatureUnit	false		
distanceUnit	Common.DistanceUnit	false		

RadioControlCapabilities

NAME	TYPE	MANDATORY	ADDITIONAL	DESCRIPTION
moduleName	String	true	maxlength: 100	The short friendly name of the climate control module. It should not be used to identify a module by mobile application.
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.

NAME	TYPE	MANDATORY	ADDITIONAL	DESCRIPTION
radioFrequencyAvailable	Boolean	false		Availability of the control of radio frequency. True: Available, False: Not Available, Not present: Not Available.
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.

NAME	TYPE	MANDATORY	ADDITIONAL	DESCRIPTION
availableHDsAvailable	Boolean	false		Availability of the getting the number of available HD channels. True: Available, False: Not Available, Not present: Not Available.
stateAvailable	Boolean	false		Availability of the getting the Radio state. True: Available, False: Not Available, Not present: Not Available.

NAME	TYPE	MANDATORY	ADDITIONAL	DESCRIPTION
signalStrengthAvailable	Boolean	false		Availability of the getting the signal strength. True: Available, False: Not Available, Not present: Not Available.
signalChangeThresholdAvailable	Boolean	false		Availability of the getting the signal Change Threshold. True: Available, False: Not Available, Not present: Not Available.

NAME	TYPE	MANDATORY	ADDITIONAL	DESCRIPTION
sisDataAvailable	Boolean	false		Availability of the getting HD radio Station Information Service (SIS) data. True: Available, False: Not Available, Not present: Not Available.
hdRadioEnableAvailable	Boolean	false		Availability of the control of enable/ disable HD radio. True: Available, False: Not Available, Not present: Not Available.

NAME	TYPE	MANDATORY	ADDITIONAL	DESCRIPTION
siriusxmRadioAvailable	Boolean	false		Availability of sirius XM radio. True: Available, False: Not Available, Not present: Not Available.

ExternalConsentStatus

NAME	TYPE	MANDATORY	ADDITIONAL	DESCRIPTION
entityType	Integer	true	minvalue: 0 maxvalue: 128	The entityType which status is informed by "status" param.
entityID	Integer	true	minvalue: 0 maxvalue: 128	The corresponding ID of entityType which status is informed by "status" param.
status	Common.EntityStatus	true	-	Status of External User Consent Settings entity: "ON" or "OFF"

Rectangle

NAME	TYPE	MANDATORY	ADDITIONAL	DESCRIPTION
x	Float	true		The X-coordinate of the rectangle
y	Float	true		The Y-coordinate of the rectangle
width	Float	true		The width of the rectangle
height	Float	true		The height of the rectangle

HapticRect

NAME	TYPE	MANDATORY	ADDITIONAL	DESCRIPTION
id	Integer	true	minvalue: 0 maxvalue: 128	A unique identifier for the haptic rectangle
rect	Common.Rectangle	true		The position of the haptic rectangle to be highlighted. The center of this rectangle is considered "touched" when the element is focused and then selected.

FuelRange

NAME	TYPE	MANDATORY	ADDITIONAL	DESCRIPTION
type	FuelType	false		
range	Float	faslse	minvalue=0 maxvalue=10000	The estimate range in KM the vehicle can travel based on fuel level and consumption

MassageModeData

NAME	TYPE	MANDATORY	ADDITIONAL	DESCRIPTION
massageZone	Common.MassageZone	true		
massageMode	Common.MassageMode	true		

MassageCushionFirmness

NAME	TYPE	MANDATORY	ADDITIONAL	DESCRIPTION
cushion	Common.MassageCushion	true		
firmness	Integer	true	minvalue="0" maxvalue="100"	

SeatMemoryAction

NAME	TYPE	MANDATORY	ADDITIONAL	DESCRIPTION
id	Integer	true	minvalue="1" maxvalue="10"	
label	String	false	maxlength="100"	
action	Common.SeatMemoryActionType	true		

SeatControlData

NAME	TYPE	MANDATORY	ADDITIONAL	DESCRIPTION
id	Common.SupportedSeat	true		
heatingEnabled	Boolean	false		
coolingEnabled	Boolean	false		
heatingLevel	Integer	false	minvalue="0" maxvalue="100"	
coolingLevel	Integer	false	minvalue="0" maxvalue="100"	
horizontalPosition	Integer	false	minvalue="0" maxvalue="100"	
verticalPosition	Integer	false	minvalue="0" maxvalue="100"	
frontVerticalPosition	Integer	false	minvalue="0" maxvalue="100"	
backVerticalPosition	Integer	false	minvalue="0" maxvalue="100"	
backTiltAngle	Integer	false	minvalue="0" maxvalue="100"	
headSupportHorizontalPosition	Integer	false	minvalue="0" maxvalue="100"	

NAME	TYPE	MANDATORY	ADDITIONAL	DESCRIPTION
headSupportVerticalPosition	Integer	false	minvalue="0" maxvalue="100"	
massageEnabled	Boolean	false		
massageMode	Common.MassageModeData	true	minsize="1" maxsize="2" array="true"	
massageCushionFirmness	Common.MassageCushionFirmness	false	minsize="1" maxsize="5" array="true"	
memory	Common.SeatMemoryAction	false		

SeatControlCapabilities

NAME	TYPE	MANDATORY	ADDITIONAL	DESCRIPTION
moduleName	String	true	maxlength="100"	The short friendly name of the light control module. It should not be used to identify a module by mobile application.
heatingEnabledAvailable	Boolean	false		
coolingEnabledAvailable	Boolean	false		
heatingLevelAvailable	Boolean	false		
coolingLevelAvailable	Boolean	false		
horizontalPositionAvailable	Boolean	false		
verticalPositionAvailable	Boolean	false		
frontVerticalPositionAvailable	Boolean	false		
backVerticalPositionAvailable	Boolean	false		
backTiltAngleAvailable	Boolean	false		

NAME	TYPE	MANDATORY	ADDITIONAL	DESCRIPTION
headSupportHorizontalPositionAvailable	Boolean	false		
headSupportVerticalPositionAvailable	Boolean	false		
massageEnabledAvailable	Boolean	false		
massageModeAvailable	Boolean	false		
massageCushionFirmnessAvailable	Boolean	false		
memoryAvailable	Boolean	false		

DateTime

NAME	TYPE	MANDATORY	ADDITIONAL	DESCRIPTION
millisecond	Integer	false	minvalue: 0 maxvalue: 999	Milliseconds – part of time - one thousandth split second
seconds	Integer	false	minvalue: 0 maxvalue: 60	Seconds part of time
minutes	Integer	false	minvalue: 0 maxvalue: 59	Minutes part of time
hour	Integer	false	minvalue: 0 maxvalue: 23	Hours part of time. Note that this structure accepts time only in 24 Hr format
day	Integer	false	minvalue: 1 maxvalue: 31	Day of the month
month	Integer	false	minvalue: 1 maxvalue: 12	Month of the year
year	Integer	false	minvalue: 0 maxvalue: 4095	The year in YYYY format
tz_hour	Integer	false	minvalue: -12 maxvalue: 14 defvalue: 0	Time zone offset in Hours with regard to UTC.
tz_minute	Integer	false	minvalue: 0 maxvalue: 59 defvalue: 0	Time zone offset in Min with regard to UTC.

Coordinate

NAME	TYPE	MANDATORY	ADDITIONAL	DESCRIPTION
latitudeDegrees	Float	true	minvalue: -90 maxvalue: 90	Latitude of the location
longitudeDegrees	Float	true	minvalue: -180 maxvalue: 180	Longitude of the location

OASISAddress

NAME	TYPE	MANDATORY	ADDITIONAL	DESCRIPTION
countryName	String	false	minlength: 0 maxlength: 200	Name of the country (localized)
countryCode	String	false	minlength: 0 maxlength: 50	Name of country (ISO 3166-2)
postalCode	String	false	minlength: 0 maxlength: 16	(PLZ, ZIP, PIN, CAP etc.)
administrativeArea	String	false	minlength: 0 maxlength: 200	Portion of country (e.g. state)
subAdministrativeAre	String	false	minlength: 0 maxlength: 200	Portion of administrativeArea (e.g. county)
locality	String	false	minlength: 0 maxlength: 200	Hypernym for city/ village
subLocality	String	false	minlength: 0 maxlength: 200	Hypernym for district
thoroughfare	String	false	minlength: 0 maxlength: 200	Hypernym for street, road etc
subThoroughfare	String	false	minlength: 0 maxlength: 200	Portion of thoroughfare (e.g. house number)

LocationDetails

NAME	TYPE	MANDATORY	ADDITIONAL	DESCRIPTION
coordinate	Common.Coordinate	false		Latitude/ Longitude of the location
locationName	String	false	maxlength: 500	Name of location
addressLines	String	false	maxlength: 500 array: true minsize: 0 maxsize: 4	Location address for display purposes only
locationDescription	String	false	maxlength: 500	Description intended location/ establishment (if applicable)
phoneNumber	String	false	maxlength: 500	Phone number of location/ establishment
locationImage	Common.Image	false		Image/icon of intended location
searchAddress	Common.OASISAddress	false		Address to be used by navigation engines for search

SyncMsgVersion

NAME	TYPE	MANDATORY	ADDITIONAL	DESCRIPTION
majorVersion	Integer	true	minvalue: 1 maxvalue: 10	The major version indicates versions that is not-compatible to previous versions
minorVersion	Integer	true	minvalue: 0 maxvalue: 1000	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	minvalue: 0 maxvalue: 1000	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

AppServiceManifest

NAME	TYPE	MANDATORY	ADDITIONAL	DESCRIPTION
serviceName	String	false		Unique name of this service
serviceType	String	true		The type of service that is to be offered by this app
serviceIcon	Common.Image	false		The icon to be associated with this service. Most likely the same as the appIcon
allowAppConsumers	Boolean	false	defvalue: 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

NAME	TYPE	MANDATORY	ADDITIONAL	DESCRIPTION
rpcSpecVersion	Common.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	array: true	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	Common.MediaServiceManifest	false		

NAME	TYPE	MANDATORY	ADDITIONAL	DESCRIPTION
weatherServiceManifest	Common.WeatherServiceManifest	false		
navigationServiceManifest	Common.NavigationServiceManifest	false		

AppServiceRecord

NAME	TYPE	MANDATORY	ADDITIONAL	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	Common.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

NAME	TYPE	MANDATORY	ADDITIONAL	DESCRIPTION
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

NAME	TYPE	MANDATORY	ADDITIONAL	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		A unique ID tied to this specific service
mediaServiceData	Common.MediaServiceData	false		
weatherServiceData	Common.WeatherServiceData	false		
navigationServiceData	Common.NavigationServiceData	false		

AppServiceCapability

NAME	TYPE	MANDATORY	ADDITIONAL	DESCRIPTION
updateReason	Common.ServiceUpdateReason	false		Only included in OnSystemCapabilityUpdated. Update reason for service record
updatedAppServiceRecord	Common.AppServiceRecord	true		Service record for a specific app service provider

AppServicesCapabilities

NAME	TYPE	MANDATORY	ADDITIONAL	DESCRIPTION
appServices	Common.AppServiceCapability	false	array: true	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

NAME	TYPE	MANDATORY	ADDITIONAL	DESCRIPTION
systemCapabilityType	Common.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	Common.NavigationCapability	false		Describes extended capabilities for onboard navigation system
phoneCapability	Common.PhoneCapability	false		Describes extended capabilities of the module's phone feature
videoStreamingCapability	Common.VideoStreamingCapability	false		Describes extended capabilities of the module's phone feature

NAME	TYPE	MANDATORY	ADDITIONAL	DESCRIPTION
remoteControlCapability	Common.RemoteControlCapabilities	false		Describes extended capabilities of the module's phone feature
appServicesCapabilities	Common.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

MediaServiceManifest

There are no defined parameters for this struct.

MediaServiceData

NAME	TYPE	MANDATORY	ADDITIONAL	DESCRIPTION
mediaType	Common.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

NAME	TYPE	MANDATORY	ADDITIONAL	DESCRIPTION
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

NAME	TYPE	MANDATORY	ADDITIONAL	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

NAME	TYPE	MANDATORY	ADDITIONAL	DESCRIPTION
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

NAME	TYPE	MANDATORY	ADDITIONAL	DESCRIPTION
trackPlaybackDuration	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

NAME	TYPE	MANDATORY	ADDITIONAL	DESCRIPTION
queuePlaybackProgress	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

NAME	TYPE	MANDATORY	ADDITIONAL	DESCRIPTION
queuePlaybackDuration	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

NAME	TYPE	MANDATORY	ADDITIONAL	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)

NAME	TYPE	MANDATORY	ADDITIONAL	DESCRIPTION
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)

WeatherServiceManifest

NAME	TYPE	MANDATORY	ADDITIONAL	DESCRIPTION
currentForecastSupported	Boolean	false		
maxMultidayForecastAmount	Integer	false		
maxHourlyForecastAmount	Integer	false		
maxMinutelyForecastAmount	Integer	false		
weatherForLocationSupported	Boolean	false		

WeatherAlert

NAME	TYPE	MANDATORY	ADDITIONAL	DESCRIPTION
title	String	false		
summary	String	false		
expires	Common.DateTime	false		
regions	String	false	array: true minsize: 1 maxsize: 99	
severity	String	false		
timeIssued	Common.DateTime	false		

WeatherData

NAME	TYPE	MANDATORY	ADDITIONAL	DESCRIPTION
currentTemperature	Common.Temperature	false		
temperatureHigh	Common.Temperature	false		
temperatureLow	Common.Temperature	false		
apparentTemperature	Common.Temperature	false		
apparentTemperatureHigh	Common.Temperature	false		
apparentTemperatureLow	Common.Temperature	false		
weatherSummary	String	false		
time	Common.DateTime	false		
humidity	Float	false	minvalue: 0 maxvalue: 1	0 to 1, percentage humidity
cloudCover	Float	false	minvalue: 0 maxvalue: 1	0 to 1, percentage cloud cover
moonPhase	Float	false	minvalue: 0 maxvalue: 1	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

NAME	TYPE	MANDATORY	ADDITIONAL	DESCRIPTION
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
precipProbability	Float	false	minvalue: 0 maxvalue: 1	0 to 1, percentage chance
precipType	String	false		e.g. "rain", "snow", "sleet", "hail"
visibility	Float	false		In km
weatherIcon	Common.Image	false		

WeatherServiceData

NAME	TYPE	MANDATORY	ADDITIONAL	DESCRIPTION
location	Common.LocationDetails	true		
currentForecast	Common.WeatherData	false		
minuteForecast	Common.WeatherData	false	array: true minsize: 15 maxsize: 60	
hourlyForecast	Common.WeatherData	false	array: true minsize: 1 maxsize: 96	
multidayForecast	Common.WeatherData	false	array: true minsize: 1 maxsize: 30	
alerts	Common.WeatherAlert	false	array: true minsize: 1 maxsize: 10	This array should be ordered with the first object being the current day

NavigationServiceManifest

NAME	TYPE	MANDATORY	ADDITIONAL	DESCRIPTION
acceptsWayPoints	Boolean	false		Informs the subscriber if this service can actually accept way points

NavigationInstruction

NAME	TYPE	MANDATORY	ADDITIONAL	DESCRIPTION
locationDetails	Common.LocationDetails	true		
action	Common.NavigationAction	true		
eta	Common.DateTime	false		
bearing	Integer	false	minvalue: 0 maxvalue:359	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	Common.NavigationJunction	false		

NAME	TYPE	MANDATORY	ADDITIONAL	DESCRIPTION
drivingSide	Common.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
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

NAME	TYPE	MANDATORY	ADDITIONAL	DESCRIPTION
image	Common.Image	false		An image representation of this instruction

NavigationServiceData

NAME	TYPE	MANDATORY	ADDITIONAL	DESCRIPTION
timeStamp	Common.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	Common.LocationDetails	false		
destination	Common.LocationDetails	false		
destinationETA	Common.DateTime	false		
instructions	Common.NavigationInstruction	false	array: true	This array should be ordered with all remaining instructions. The start of this array should always contain the next instruction
nextInstructionETA	Common.DateTime	false		

NAME	TYPE	MANDATORY	ADDITIONAL	DESCRIPTION
nextInstructionDistance	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
nextInstructionDistanceScale	Float	false		Distance till next maneuver (starting from) from previous maneuver

NAME	TYPE	MANDATORY	ADDITIONAL	DESCRIPTION
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

Low Voltage

Type

Signal
Sender HMI
Purpose To suspend/restore SDL's services due to a low voltage event

Description

A 'LowVoltage' event occurs on HMI when battery voltage hits below a certain predefined threshold set by the system.
In case of such event, SDL operations are halted including receiving and processing of normal RPC messages from HMI.

HMI sends system signals of the range SIGRTMIN - SIGRTMAX for LowVoltage, WakeUp and IgnitionOff to all smartDeviceLinkCore processes.

MUST

1. Send `LOW_VOLTAGE` signal to SDL when a 'LowVoltage' event occurs
2. Send `WAKE_UP` signal to SDL when the voltage recovers

3. Send `IGNITION_OFF` signal when SDL processes have to be terminated, applications have to be unregistered due SDL functionality suspension during a LOW VOLTAGE event

SDL BEHAVIOR IN CASE OF A `LOW_VOLTAGE` EVENT:

- SDL ignores all the requests from mobile applications without providing any kind of response.
- SDL ignores all responses and messages from HMI except for `WAKE_UP` or `IGNITION_OFF` signals.
- SDL stops audio/video streaming services.
- All transports are unavailable for SDL.
- SDL persists resumption related data stored before receiving a `LOW_VOLTAGE` signal.
- SDL and the PoliciesManager must persist 'consumer data' (resumption-related + local PT).

After the voltage level is restored HMI sends `WAKE_UP` signal to SDL and SDL processes its regular work and all operations resumption.

SDL BEHAVIOR AFTER RECEIVING A "WAKE_UP" SIGNAL:

- After receiving a `WAKE_UP` signal, all applications will be unregistered and the device disconnected.
- If `LOW_VOLTAGE` was received at the moment of writing to policies database, SDL and Policies Manager must keep policies database correct and working. After `WAKE_UP` policy database reflects the last known correct state.
- SDL must be able to start up correctly in the next ignition cycle after it was powered off in low voltage state.

DETAILS OF IMPLEMENTATION

UNIX signals are used to exchange shutdown and wake-up signals between HMI and SDL Core during a 'LowVoltage' event.

UNIX signals provide ability to use signals from SIGRTMIN to SIGRTMAX for custom needs.
SDL uses this range for handling `LOW_VOLTAGE`, `WAKE_UP`, `IGNITION_OFF` notifications.

Offset for SIGRTMIN from this notifications is defined in the 'Main' section of the smartDeviceLink.ini file:

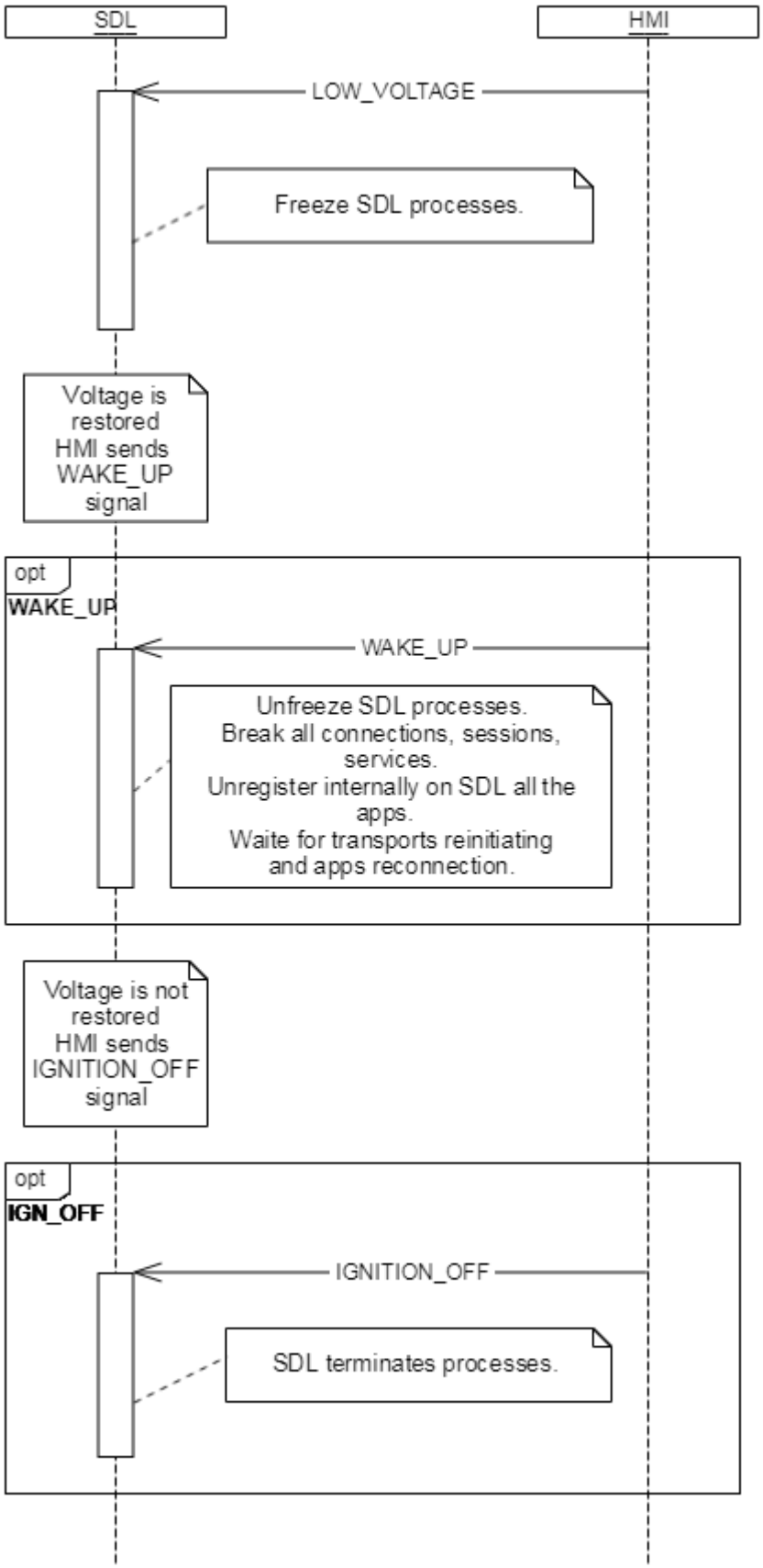
```
[MAIN]
LowVoltageSignal = 1 ; Offset for from SIGRTMIN
WakeUpSignal = 2 ; Offset for from SIGRTMIN
IgnitionOffSignal = 3 ; Offset for from SIGRTMIN
```

Sequence Diagrams

SEQUENCE DIAGRAM

Low Voltage

View Diagram


SDL
HMI
LOW_VOLTAGE
Freeze SDL processes.
Voltage is restored HMI sends WAKE_UP signal
opt
WAKE_UP
WAKE_UP
Unfreeze SDL processes.
Break all connections, sessions, services.
Unregister internally on SDL all the apps.
Waite for transports reinitiating and apps reconnection.
Voltage is not restored HMI sends IGNITION_OFF signal
opt
IGN_OFF
IGNITION_OFF
SDL terminates processes.