

SDL Integration Guidelines

Table of Contents

Table of Contents	2
Introduction.....	44
Abbreviations and Definitions.....	45
I. SDL Programmer` s Guide.....	47
1 Overview.....	47
2 Architecture	47
3 Transport Manager	48
3.1 General description	48
3.1.1 Transport layer features.....	48
3.1.2 Transport Level Structure	50
3.1.2.1 Transport Manager Structure	51
3.1.2.2 Transport Adapter Structure	52
3.1.3 Operation Examples (Message Sequence Chart) 53	
3.1.3.1 New device search	53
3.1.3.2 Device-originating connection	55
3.1.3.3 Connection close command	56
3.2 Transport Manager (TM) Usage	57
3.2.1 General Processing Description	57
3.2.1.1 TM initialization	57
3.2.1.2 TM structure.....	58
3.2.1.3 TA initialization.....	58
3.2.1.4 Getting started	58
3.2.1.5 Errors in TM	59
3.2.1.6 Messages in TM	59
3.2.1.7 Connection identifiers	59
3.2.1.8 Connection closing	60
3.2.1.9 SDL shutting down	61
3.2.2 Create Default TM Instance.....	61
3.2.3 Add Custom Listener to TM	61
3.2.4 Add Custom Transport Adapter to Default TM 62	
3.2.4.1 About TA in general.....	62
3.2.4.2 Several instances of TA.....	62
3.2.4.3 Add Custom TA	63
3.2.4.4 Initialization.....	63
3.2.5 Custom TA implementation from scratch.....	64
3.2.6 Add a New Listener to TA.....	65

3.2.7	Create TM with Custom TAs Only (with No Default Adapter).....	66
3.2.8	Transport Manager Customizing	67
3.2.8.1	Basic information	67
3.2.8.2	Queues as a fundamental of TM	68
3.2.8.3	Rules for developer	68
3.2.9	Transport Adapter Customizing	68
3.2.9.1	Basic information	68
3.2.9.2	Workers of TA.....	68
3.2.9.3	Connection	70
3.2.9.4	Descriptor	70
3.2.9.5	Create custom Transport Adapter	70
4	Build and Run SDL and Set Up the Environment.....	72
4.1	General.....	72
4.2	Known issues	73
4.3	Preparation steps	73
4.3.1	To prepare the Linux Host	73
4.3.2	To set up Android Simulator for using instead of real device	74
5	Audio/Video Streaming over SDL.....	77
5.1	PulseAudio and GStreamer libraries: how to include or exclude from project	77
5.1.1	Short overview	77
5.1.2	Include	77
5.1.3	Exclude	77
5.2	Use SyncProxyTester to check audio/video streaming over SDL	77
5.3	Configuring audio/video streaming parameters in smartDeviceLink.ini file.....	78
II	SDL-HMI Integration Guide	80
1	Overview.....	80
1.1	Operation System.....	80
1.2	Transports Supported.....	80
1.3	Communication.....	81
2	WebSocket Transport.....	81
2.1	Connection Opening	81
2.1.1	Requirements to HMI adapter.....	81
2.1.2	Handshake.....	81
2.1.3	Components registering.....	82
2.1.4	Examples	82
2.2	JSON Message Format.....	86
2.2.1	Request	86

2.2.1.1 Examples	87
2.2.2 Notification	88
2.2.2.1 Examples	88
2.2.3 Response.....	89
2.2.3.1 Examples	90
2.2.4 Error.....	91
2.2.4.1 Examples	92
5 Getting Started	93
5.1 Readiness confirming.....	93
5.1.1 Sequence diagram.....	93
6 BasicCommunication Component Description	95
6.1 UpdateDeviceList	95
6.1.1 Description.....	95
6.1.2 Request	96
6.1.2.1 Behavior	96
6.1.2.2 Parameters.....	96
6.1.2.3 DeviceInfo Structure.....	96
6.1.2.4 TransportType Enumeration	97
6.1.3 Response.....	97
6.1.4 Sequence Diagrams	98
6.1.4.1 UpdateDeviceList after SDL finds a new device over BT and USB	98
6.1.4.2 UpdateDeviceList (TransportType=USB_IOS).....	99
6.1.4.3 UpdateDeviceList (TransportType=USB_AOA)	100
6.1.5 JSON Messages Examples	100
6.1.5.1 Request.....	100
6.1.5.2 Response	101
6.1.5.3 Error message.....	101
6.2 ActivateApp	101
6.2.1 Description.....	101
6.2.2 Request	102
6.2.2.1 Behavior	102
6.2.2.2 Parameters.....	103
6.2.2.3 AppPriority.....	103
6.2.2.4 HMILevel	103
6.2.3 Response.....	104
6.2.4 Sequence Diagrams	106
6.2.4.1 BasicCommunication.ActivateApp after successsful resumption.....	106
6.2.4.2 BasicCommunication.ActivateApp after UNsuccesssful resumption.....	109

6.2.4.3 BasicCommunication.ActivateApp after unexpected disconnect	111
6.2.4.4 BasicCommunication.ActivateApp after the User accepted the data consent prompt.....	112
6.2.4.5 BasicCommunication.ActivateApp after the User declined the data consent prompt	113
6.2.5 JSON Messages Examples	114
6.2.5.1 Request.....	114
6.2.5.2 Response	114
6.2.5.3 Error message.....	114
6.3 MixingAudioSupported	114
6.3.1 Description.....	114
6.3.2 Request	115
6.3.2.1 Behavior	115
6.3.3 Response.....	115
6.3.3.1 Behavior	115
6.3.3.2 Parameters.....	115
6.3.4 Sequence Diagrams	115
6.3.4.1 MixingAudioSupported Messaging.....	115
6.3.5 JSON Messages Examples	116
6.3.5.1 Request.....	116
6.3.5.2 Response	116
6.3.5.3 Error message.....	116
6.4 AllowDeviceToConnect	117
6.4.1 Description.....	117
6.4.2 Request	117
6.4.2.1 Behavior	117
6.4.2.2 Parameters.....	117
6.4.2.3 DeviceInfo Structure.....	117
6.4.2.4 TransportType Enumeration	118
6.4.3 Response.....	118
6.4.3.1 Behavior	118
6.4.3.2 Parameters.....	119
6.4.4 Sequence Diagrams	119
6.4.4.1 AllowDeviceToConnect Messaging.....	119
6.4.5 JSON Messages Examples	119
6.4.5.1 Request.....	119
6.4.5.2 Response	119
6.4.5.3 Error message.....	120
6.5 SystemRequest.....	120
6.5.1 Description.....	120
6.5.2 Request	120

6.5.2.1 Behavior	120
6.5.2.2 Parameters.....	121
6.5.2.3 RequestType	121
6.5.3 Response.....	121
6.5.4 Sequence Diagrams	123
6.5.4.1 SystemRequest workflow	123
6.5.5 JSON Messages Examples	123
6.5.5.1 Request.....	123
6.5.5.2 Response	123
6.5.5.3 Error message.....	124
6.6 GetSystemInfo	124
6.6.1 Description.....	124
6.6.2 Request	124
6.6.2.1 Behavior.....	124
6.6.3 Response.....	124
6.6.3.1 Parameters.....	125
6.6.3.2 Language Enumeration.....	125
6.6.4 Sequence Diagrams	126
6.6.4.1 GetSystemInfo	126
6.6.5 JSON Messages Examples	126
6.6.5.1 Request.....	126
6.6.5.2 Response	127
6.6.5.3 Error message.....	127
6.7 OnReady	127
6.7.1 Description.....	127
6.7.2 Sequence Diagrams	128
6.7.2.1 OnReady after WebSocket connection establishment	128
6.7.2.2 OnReady after D-Bus service publishing	128
6.7.3 JSON Messages Examples	128
6.8 OnStartDeviceDiscovery	129
6.8.1 Description.....	129
6.8.2 Sequence Diagrams	129
6.8.2.1 OnStartDeviceDiscovery upon User`s request and resulting UpdateDeviceList	129
6.8.3 JSON Messages Examples	130
6.9 OnDeviceChosen	130
6.9.1 Description.....	130
6.9.1.1 Parameters.....	131
6.9.1.2 DeviceInfo Structure.....	131
6.9.1.3 TransportType Enumeration	131

6.9.2 Sequence Diagrams	132
6.9.2.1 OnDeviceChosen with preceding UpdateDeviceList (BT transport).....	132
6.9.3 JSON Messages Examples	132
6.10 OnFindApplications	133
6.10.1 Description.....	133
6.10.1.1 Parameters.....	133
6.10.1.2 DeviceInfo Structure.....	133
6.10.1.3 TransportType Enumeration	134
6.10.2 Sequence Diagrams	134
6.10.2.1 OnFindApplications upon User's choice and resulting OnAppRegistered	134
6.10.2.2 OnFindApplications upon User's choosing the device in the list of found devices and resulting OnAppRegistered.....	135
6.10.3 JSON Messages Examples	135
6.11 OnAppActivated	136
6.11.1 Description.....	136
6.11.1.1 Parameters.....	136
6.11.2 Sequence Diagrams	137
6.11.2.1 OnAppActivated upon User's choice and resulting ActivateApp	137
6.11.3 JSON Messages Examples	137
6.12 OnAppDeactivated	138
6.12.1 Description.....	138
6.12.1.1 Parameters.....	139
6.12.1.2 DeactivateReason Enumeration	139
6.12.2 Sequence Diagrams	139
6.12.2.1 OnAppDeactivated Simple Messaging Example.....	139
6.12.3 JSON Messages Examples	140
6.13 OnAppRegistered.....	140
6.13.1 Description.....	140
6.13.1.1 Parameters.....	143
6.13.1.2 HMIApplication Structure	143
6.13.1.3 AppHMIType Enumeration	144
6.13.1.4 AppPriority Enumeration	145
6.13.1.5 RequestType.....	145
6.13.1.6 DeviceInfo Structure	146
6.13.1.7 TransportType Enumeration	146
6.13.2 Sequence Diagrams	146
6.13.2.1 OnAppRegistered with resume=true after unexpected disconnect	147

6.13.2.2 OnAppRegistered with 'resume'=false parameter after unexpected disconnect.....	150
6.13.2.3 OnAppRegistered USB transport connection.....	151
6.13.2.4 OnAppRegistered Bluetooth transport connection	151
6.13.3 JSON Messages Examples	151
6.14 OnAppUnregistered	152
6.14.1 Description.....	152
6.14.1.1 Parameters.....	153
6.14.2 Sequence Diagrams	153
6.14.2.1 OnAppUnregistered of active application.....	153
6.14.3 JSON Messages Examples	154
6.15 OnExitApplication.....	154
6.15.1 Description.....	154
6.15.1.1 Parameters.....	155
6.15.1.2 ApplicationExitReason	155
6.15.2 Sequence Diagrams	156
6.15.2.1 OnExitApplication USER_EXIT	156
6.15.2.2 OnExitApplication UNAUTHORIZED_TRANSPORT_REGISTRATION.....	157
6.15.3 JSON Messages Examples	157
6.16 OnExitAllApplications	157
6.16.1 Description.....	157
6.16.1.1 Parameters.....	158
6.16.1.2 ApplicationsCloseReason Enumeration.....	158
6.16.2 Sequence Diagrams	158
6.16.2.1 OnExitAllApplications SUSPEND and IGNITION OFF	158
6.16.3 JSON Messages Examples	159
6.17 PlayTone	159
6.18.1 Description.....	159
6.17.2 Request	160
6.17.2.1 Parameters.....	160
6.17.2.2 MethodName.....	160
6.17.3 Response.....	160
6.17.4 Sequence Diagrams	161
6.17.4.1 PlayTone with Alert request	161
6.17.5 JSON Messages Examples	161
6.17.5.1 Request.....	161
6.17.5.2 Response	161
6.17.5.3 Error message.....	162
6.18 OnSystemRequest	162

6.18.1 Description	162
6.18.1.1 Parameters.....	163
6.18.1.2 RequestType Enumeration	163
6.18.1.3 FileType Enumeration	164
6.18.2 Sequence Diagrams	164
6.18.2.1 OnSystemRequest requests the file to download from Cloud	164
6.18.3 JSON Messages Examples	164
6.18.4 D-Bus Messages Examples.....	165
6.19 OnPutFile	165
6.19.1 Description.....	165
6.19.1.1 Parameters.....	166
6.19.1.2 FileType Enumeration	166
6.19.2 Sequence Diagrams	167
6.19.2.1 OnPutFile. File is uploaded before RPC is used.....	167
6.19.2.2 OnPutFile. File is uploaded after RPC is used	168
6.19.2.3 OnPutFile. File uploading via SystemRequest.....	169
6.19.3 JSON Messages Examples	169
6.20 OnFileRemoved	170
6.20.1 Description.....	170
6.20.1.1 Parameters.....	170
6.20.1.2 FileType Enumeration	170
6.20.2 Sequence Diagrams	171
6.20.2.1 OnFileRemoved	171
6.20.3 JSON Messages Examples	171
6.21 OnSDLClose	172
6.21.1 Description.....	172
6.21.2 Sequence Diagrams	173
6.21.2.1 OnSDLClose IGNITION_OFF	173
6.21.2.1 OnSDLClose MASTER_RESET	173
6.21.3 JSON Messages Examples	174
6.22 OnUpdateDeviceList	174
6.22.1 Description.....	174
6.22.2 Sequence Diagrams	175
6.22.2.1 OnUpdateDeviceList upon User's request and resulting UpdateDeviceList	175
6.22.3 JSON Messages Examples	175
6.23 OnResume AudioSource	176
6.23.1 Description	176
6.23.1.1 Parameters.....	176
6.23.2 Sequence Diagrams	177

6.23.2.1 AudioSource resumption for the app with FULL HMILevel	177
6.23.2.2 OnResumeAudioSource for the app with LIMITED HMILevel	179
6.23.2.3 OnResumeAudioSource for the app with LIMITED HMILevel (Phone call is active).....	181
6.23.2.4 AudioSource resume for the app_1 with FULL HMILevel and app_2 just registered while Phone call is active.....	183
6.23.3 JSON Messages Examples	184
6.24 UpdateAppList.....	184
6.24.1 Description.....	184
6.24.2 Request	184
6.24.2.1 Behavior	184
6.24.2.2 Parameters.....	184
6.24.2.3 HMIApplication Structure	185
6.24.2.4 AppHMIType Enumeration	186
6.24.3 Response.....	186
6.24.4 Sequence Diagrams	186
6.24.4.1 UpdateAppList after application has just registered with SDL	186
6.24.4.2 UpdateAppList on User's Request (BT transport).....	187
6.24.5 JSON Messages Examples	188
6.24.5.1 Request.....	188
6.24.5.2 Response	189
6.24.5.3 Error message.....	189
6.25 OnSDLPersistenceComplete	189
6.25.1 Description.....	189
6.25.2 Sequence Diagrams	190
6.25.2.1 OnSDLPersistenceComplete	190
6.25.3 JSON Messages Examples	191
6.26 OnPhoneCall.....	191
6.26.1 Description.....	191
6.26.1.1 Parameters.....	191
6.26.2 Sequence Diagrams	192
6.26.2.1 OnPhoneCall	192
6.26.3 JSON Messages Examples	192
6.27 OnEmergencyEvent.....	193
6.27.1 Description.....	193
6.27.1.1 Parameters.....	193
6.27.2 Sequence Diagrams	194
6.27.2.1 OnEmergencyEvent	194

6.27.3 JSON Messages Examples	194
6.28 OnAwakeSDL.....	195
6.28.1 Description.....	195
6.28.2 Sequence Diagrams	196
6.28.2.1 OnAwakeSDL.....	196
6.28.3 JSON Messages Examples	196
6.29 DialNumber	196
6.29.1 Description.....	196
6.29.2 Request	197
6.29.2.1 Behavior.....	197
6.29.2.1 Parameters	197
6.29.3 Response.....	197
6.29.4 Sequence Diagrams	199
6.29.5.1 DialNumber Success.....	199
6.29.5.1 DialNumber Failed	200
6.29.5 JSON Messages Examples	200
6.29.5.1 Request.....	200
6.29.5.2 Response	200
6.29.5.3 Error message.....	200
7 UI Component Description	201
7.1 IsReady	201
7.1.1 Description.....	201
7.1.2 Request	201
7.1.2.1 Behavior.....	201
7.1.3 Response.....	201
7.1.3.1 Parameters.....	202
7.1.4 Sequence Diagrams	202
7.1.4.1 UI.IsReady and preceding OnReady	202
7.1.5 JSON Messages Examples	202
7.1.5.1 Request.....	202
7.1.5.2 Response	202
7.1.5.3 Error message.....	203
7.2 GetCapabilities	203
7.2.1 Description.....	203
7.2.2 Request	203
7.2.2.1 Behavior.....	203
7.2.3 Response.....	204
7.2.3.1 Parameters.....	204
7.2.3.2 DisplayCapabilities Structure	205
7.2.3.3 DisplayType Enumeration	205

7.2.3.4 TextFieldName Enumeration	206
7.2.3.5 ImageField Structure.....	209
7.2.3.6 ImageFieldName Enumeration	210
7.2.3.7 FileType Enumeration	210
7.2.3.8 ImageResolution Structure.....	211
7.2.3.9 MediaClockFormat Enumeration.....	211
7.2.3.10 ImageType Enumeration	212
7.2.3.11 ScreenParams Structure	212
7.2.3.12 TouchEventCapabilities Structure	212
7.2.3.13 SoftButtonCapabilities Structure	212
7.2.3.14 HmiZoneCapabilities Enumeration.....	213
7.2.3.15 AudioPassThruCapabilities Structure	213
7.2.3.16 SamplingRate Enumeration	214
7.2.3.17 BitsPerSample Enumeration	214
7.2.3.18 AudioType Enumeration.....	214
7.2.3.19 HMICapabilities structure	214
7.2.4 Sequence Diagrams	214
7.2.4.1 UI.GetCapabilities and preceding IsReady	214
7.2.5 JSON Messages Examples	215
7.2.5.1 Request.....	215
7.2.5.2 Response	215
7.2.5.3 Error message.....	216
7.3 GetSupportedLanguages	216
7.3.1 Description.....	216
7.3.2 Request	217
7.3.2.1 Behavior	217
7.3.3 Response.....	217
7.3.3.1 Parameters.....	218
7.3.3.2 Language Enumeration.....	218
7.3.4 Sequence Diagrams	219
7.1.4.1 GetSupportedLanguages with preceding IsReady	219
7.3.5 JSON Messages Examples	219
7.3.5.1 Request.....	219
7.3.5.2 Response	219
7.3.5.3 Error message.....	220
7.4 Alert.....	220
7.4.1 Description.....	220
7.4.2 Request	221
7.4.2.1 Behavior	221
7.4.2.2 Parameters.....	223

7.4.2.3 TextFieldStruct Structure	223
7.4.2.4 TextFieldName Enumeration	223
7.4.2.4 AlertType Enumeration	224
7.4.3 Response.....	224
7.4.3.1 Parameters.....	225
7.4.4 Sequence Diagrams	226
7.4.4.1 Alert together with PlayTone and Speak RPCs and closed by the timeout	226
7.4.4.2 Alert displayed and closed by DEFAULT_ACTION soft button press.....	227
7.4.4.3 Alert for non-activa application displayed and closed by STEAL_FOCUS soft button press	228
7.4.4.4 Alert is displayed and then aborted by VR session started	229
7.4.4.5 Alert is rejected because of RPC of a higher priority currently presented on UI	229
7.4.4.6 Alert BOTH when UI is closed earlier than TTS has finished speaking.....	231
7.4.5 Possible Layout	231
7.4.5.1 Alert dialog with text fields and NO soft buttons.....	231
7.4.5.2 Alert dialog with text fields, one soft button and a progress indicator.....	232
7.4.5.3 Alert dialog with text fields and four soft buttons.....	232
7.4.6 JSON Messages Examples	232
7.4.6.1 Request.....	232
7.4.6.2 Response	233
7.4.6.3 Error message.....	233
7.5 Show	233
7.5.1 Description.....	233
7.5.2 Request	234
7.5.2.1 Behavior.....	234
7.5.2.2 Parameters.....	234
7.5.2.3 TextFieldStruct Structure	236
7.5.2.4 TextFieldName Enumeration	236
7.5.2.5 TextAlign Enumeration	237
7.5.3 Response.....	237
7.5.4 Sequence Diagrams	238
7.5.4.1 Show for the active application that is then deactivated and activated again.....	238
7.5.4.2 Show for the application currently not active on HMI.....	240
7.5.4.3 Requested with Show soft button press.....	241
7.5.4.4 Show text fields expected behavior.....	242
7.5.5 Possible Layout	244

7.5.5.1 Persistent display areas updated with Show for media applications	244
7.5.5.2 Persistent display areas updated with Show for non-media applications	244
7.5.6 JSON Messages Examples	245
7.5.6.1 Request.....	245
7.5.6.2 Response	246
7.5.6.3 Error message.....	246
7.6 AddCommand	246
7.6.1 Description.....	246
7.6.2 Request	246
7.6.2.1 Behavior.....	246
7.6.2.2 Parameters.....	248
7.6.2.3 MenuParams.....	248
7.6.3 Response.....	249
7.6.4 Sequence Diagrams	250
7.6.4.1 AddCommand for the active application on HMI, the command is chosen by the User.....	251
7.6.4.2 AddCommand for the application not active on HMI.....	252
7.6.4.3 AddCommand: adding command to sub menu.....	253
7.6.4.4 AddCommand: expected behavior of adding commands depending on position parameter	254
7.6.4.5 AddCommand rejected because of the limit of menu items exhausted	256
7.6.4.6 UI.AddCommand returns SUCCESS, VR portion failed.....	256
7.6.4.7 UI.AddCommand returns SUCCESS, VR portion no response	257
7.6.4.8 UI.AddCommand fails, VR portion SUCCESS	257
7.6.4.9 UI.AddCommand no response, VR portion SUCCESS	258
7.6.5 Possible Layout	258
7.6.5.1 Application main menu with sub menus and commands.....	258
7.6.5.2 Application sub menu with commands	259
7.6.6 JSON Messages Examples	259
7.8.6.1 Request.....	259
7.8.6.2 Response	259
7.8.6.3 Error message.....	260
7.7 DeleteCommand	260
7.7.1 Description.....	260
7.7.2 Request	260
7.8.2.1 Behavior	260

7.7.2.2 Parameters.....	261
7.7.3 Response.....	261
7.7.4 Sequence Diagrams	262
7.7.4.1 DeleteCommand, for the application active on HMI, for the command from the menu currently open on UI and preceding AddCommand	262
7.7.4.2 DeleteCommand for the application not active on HMI.....	263
7.7.5 JSON Messages Examples	265
7.7.5.1 Request.....	265
7.7.5.2 Response	265
7.7.5.3 Error message.....	265
7.8 AddSubMenu	265
7.8.1 Description.....	265
7.8.2 Request	266
7.8.2.1 Behavior	266
7.8.2.2 Parameters.....	266
7.8.2.3 MenuParams	267
7.8.3 Response.....	267
7.8.4 Sequence Diagrams	268
7.8.4.1 AddSubMenu for the application active on HMI	268
7.8.4.2 AddSubMenu for the application not active on HMI.....	269
7.8.4.3 AddSubMenu: expected behavior of adding sub menus depending on position parameter	269
7.8.4.4 AddSubMenu rejected because of the limit of menu items exhausted	271
7.8.5 Possible Layout	271
7.8.5.1 Application main menu with sub menus and commands.....	271
7.8.6 JSON Messages Examples	271
7.8.6.1 Request.....	271
7.8.6.2 Response	272
7.8.6.3 Error message.....	272
7.9 DeleteSubMenu	272
7.9.1 Description.....	272
7.9.2 Request	273
7.9.2.1 Behavior	273
7.9.2.2 Parameters.....	273
7.9.3 Response.....	274
7.9.4 Sequence Diagrams	275
7.9.4.1 DeleteSubmenu which contains the commands	276

7.9.4.2 DeleteSubmenu which is now opened on the screen	279
7.9.5 JSON Messages Examples	280
7.9.5.1 Request.....	280
7.9.5.2 Response	280
7.9.5.3 Error message.....	280
7.10 UI.PerformInteraction	280
7.10.1 Description	280
7.10.2 Request	281
7.10.2.1 Behavior	281
7.10.2.2 Parameters.....	286
7.10.2.3 TextFieldStruct Structure	287
7.10.2.4 TextFieldName	287
7.10.2.5 Choice	287
7.10.2.6 VrHelpItem	289
7.10.2.7 LayoutMode.....	289
7.10.3 Response.....	289
7.11.3 Parameters.....	290
7.10.4 Sequence Diagrams	290
7.10.4.1 UI.PerformInteraction in 'VR only' mode successfully completed	290
7.10.4.2 UI.PerformInteraction in 'Manual only' mode successfully completed	291
7.10.4.3 UI.PerformInteraction in 'Both' mode timed out	292
7.10.5 JSON Messages Examples	293
7.10.5.1 Request.....	293
7.10.5.2 Response	295
7.10.5.3 Error message.....	295
7.11 SetMediaClockTimer	295
7.11.1 Description	295
7.11.2 Request	296
7.11.2.1 Behavior	296
7.11.2.2 Parameters.....	297
7.11.2.3 TimeFormat.....	297
7.11.2.4 ClockUpdateMode.....	297
7.11.3 Response.....	298
7.11.4 Sequence Diagrams	300
7.11.4.1 SetMediaClockTimer of COUNTUP and COUNTDOWN modes for the cases of active and background applications	301
7.11.4.2 SetMediaClockTimer of PAUSE and RESUME for the application active on HMI	303

7.11.4.3 SetMediaClockTimer of COUNTDOWN for the active application that is then deactivated and activated again and SetMediaClockTimer of CLEAR for the active application	304
7.11.5 JSON Messages Examples	305
7.11.5.1 Request.....	305
7.11.5.2 Response	306
7.11.5.3 Error message.....	306
7.12 SetGlobalProperties	306
7.12.1 Description.....	306
7.12.2 Request	307
7.12.2.1 Behavior	307
7.12.2.2 Parameters.....	308
7.12.2.3 VrHelpItem	308
7.12.2.4 KeyboardProperties	308
7.12.2.5 KeyboardLayout	309
7.12.3 Response.....	309
7.12.4 Sequence Diagrams	310
7.12.4.1 SetGlobalProperties for the application active on HMI and the later VR activation	310
7.12.5 JSON Messages Examples	311
7.12.5.1 Request.....	311
7.12.5.2 Response	312
7.12.5.3 Error message.....	313
7.13 ChangeRegistration	313
7.13.1 Description.....	313
7.13.2 Request	313
7.13.2.1 Behavior	313
7.13.2.2 Parameters.....	314
7.13.2.3 Language	314
7.13.3 Response.....	315
7.14.4 Sequence Diagrams	316
7.14.4.1 ChangeRegistration	316
7.13.5 JSON Messages Examples	316
7.13.5.1 Request.....	316
7.13.5.2 Response	317
7.13.5.3 Error message.....	317
7.14 GetLanguage	317
7.14.1 Description.....	317
7.14.2 Request	318
7.14.2.1 Behavior	318
7.14.3 Response.....	318

7.14.3.1 Parameters.....	318
7.14.3.3 Language	319
7.14.4 Sequence Diagrams	319
7.14.4.1 GetLanguage	319
7.14.5 JSON Messages Examples	320
7.14.5.1 Request.....	320
7.14.5.2 Response	320
7.14.5.3 Error message.....	320
7.15 SetApplcon.....	321
7.15.1 Description.....	321
7.15.2 Request	321
7.15.2.1 Behavior	321
7.15.2.2 Parameters.....	321
7.15.2.3 Image	322
7.15.2.4 ImageType	322
7.15.3 Response.....	322
7.15.4 Sequence Diagrams	323
7.15.4.1 SetApplcon.....	323
7.15.5 Possible Layout	324
7.15.5.1 The list of registered applications with and without icons	324
7.15.6 JSON Messages Examples	325
7.15.6.1 Request.....	325
7.15.6.2 Response	325
7.15.6.3 Error message.....	325
7.16 Slider	325
7.16.1 Description.....	325
7.16.2 Request	326
7.16.2.1 Behavior	326
7.16.2.2 Parameters.....	326
7.16.3 Response.....	327
7.16.3.1 Parameters.....	328
7.16.4 Sequence Diagrams	328
7.16.4.1 Slider with static footer displayed then closed by the timeout.....	328
7.16.4.2 Slider with dynamic footer displayed then aborted	329
7.16.4.2 Slider with static footer displayed then closed by 'OK' button press	330
7.16.5 Possible Layout	331
7.16.5.1 Slider	331
7.16.5.2 Slider with dynamic footer	332

7.16.6 JSON Messages Examples	332
7.16.6.1 Request.....	332
7.16.6.2 Response	332
7.16.6.3 Error message.....	333
7.17 ScrollableMessage.....	333
7.17.1 Description.....	333
7.17.2 Request	333
7.17.2.1 Behavior	333
7.17.2.2 Parameters.....	334
7.17.2.3 TextFieldStruct Structure	335
7.17.2.4 TextFieldName Enumeration	335
7.17.3 Response.....	335
7.17.4 Sequence Diagrams	336
7.17.4.1 ScrollableMessage displayed, scrolled by the User and closed by the timeout.....	336
7.17.4.2 ScrollableMessage with soft buttons of DEFAULT_ACTION and KEEP_CONTEXT system action pressed by the User	337
7.17.4.3 ScrollableMassage with STEAL_FOCUS soft button for the application not active on HMI.....	338
7.17.5 JSON Messages Examples	339
7.17.5.1 Request.....	339
7.17.5.2 Response	340
7.17.5.3 Error message.....	341
7.18 PerformAudioPassThru.....	341
7.18.1 Description.....	341
7.18.2 Request	341
7.18.2.1 Behavior	341
7.18.2.2 Parameters.....	342
7.18.2.3 TextFieldStruct Structure	343
7.18.2.4 TextFieldName Enumeration	343
7.18.3 Response.....	343
7.18.4 Sequence Diagrams	344
7.18.4.1 PerformAudioPassTru requested together with TTS.Speak, processed and then finished by 'Retry' soft button press	345
7.18.4.2 PerformAudioPassTru for the application not active on HMI, processed and then finished by EndAudioPassThru	346
7.18.4.3 PerformAudioPassThru for the application active on HMI, SDL built with with “-DEXTENDED_MEDIA_MODE=OFF” flag (wav file reading)	347
7.18.4.4 PerformAudioPassThru for the application active on HMI, SDL built with with “-	

DEXTENDED_MEDIA_MODE=ON" flag (reading from the microphone)	348
7.18.5 JSON Messages Examples	348
7.18.5.1 Request	348
7.18.5.2 Response	349
7.18.5.3 Error message	349
7.19 EndAudioPassThru	349
7.19.1 Description	349
7.19.2 Request	349
7.19.2.1 Behavior	349
7.19.3 Response	350
7.19.4 Sequence Diagrams	350
7.19.4.1 EndAudioPassThru that stops the audio capturing	350
7.19.4.2 EndAudioPassThru for the case when audio capturing is already over	351
7.19.5 JSON Messages Examples	352
7.19.5.1 Request	352
7.19.5.2 Response	353
7.19.5.3 Error message	353
7.20 ClosePopUp	353
7.20.1 Description	353
7.20.2 Request	353
7.20.2.1 Behavior	353
7.20.2.2 Parameters	354
7.20.3 Response	354
7.20.4 Sequence Diagrams	355
7.20.4.1 ClosePopUp for UI.PerformInteraction	355
7.20.5 JSON Messages Examples	355
7.20.5.1 Request	355
7.20.5.2 Response	355
7.20.5.3 Error message	356
7.21 OnCommand	356
7.21.1 Description	356
7.21.1.1 Parameters	356
7.21.2 Sequence Diagrams	357
7.21.2.1 OnCommand	357
7.21.3 JSON Messages Examples	357
7.22 OnSystemContext	358
7.22.1 Description	358
7.22.1.1 Parameters	359
7.22.1.2 SystemContext	359

7.22.2 Sequence Diagrams	360
7.22.2.1 OnSystemContext for different HMI states.....	360
7.22.3 JSON Messages Examples	362
7.23 OnLanguageChange.....	362
7.23.1 Description.....	362
7.23.1.1 Parameters.....	362
7.23.1.2 Language	362
7.23.2 Sequence Diagrams	363
7.23.2.1 OnLanguageChange.....	363
7.23.3 JSON Messages Examples	364
7.24 OnKeyboardInput.....	365
7.24.1 Description.....	365
7.24.1.1 Parameters.....	366
7.24.1.2 KeyboardEvent.....	366
7.24.2 Sequence Diagrams	367
7.24.2.1 OnKeyboardInput for SINGLE_KEYPRESS keypressMode	367
7.24.2.2 OnKeyboardInput for QUEUE_KEYPRESSES keypressMode	369
7.24.2.3 OnKeyboardInput for RESEND_CURRENT_ENTRY keypressMode	371
7.24.2.4 OnKeyboardInput for cancelled entry.....	373
7.24.2.5 OnKeyboardInput for ENTRY_VOICE	373
7.24.3 JSON Messages Examples	374
7.25 OnTouchEvent	374
7.25.1 Description.....	374
7.25.1.1 Parameters.....	375
7.25.1.2 TouchEvent	375
7.25.1.2 TouchCoord	375
7.25.1.2 TouchType	376
7.25.2 Sequence Diagrams	377
7.25.2.1 OnTouchEvent moving two fingers stopped together	377
7.25.2.2 OnTouchEvent moving two fingers, one stopped earlier	379
7.25.3 JSON Messages Examples	381
7.26 OnResetTimeout.....	381
7.26.1 Description.....	381
7.26.1.1 Parameters.....	381
7.26.2 Sequence Diagrams	382
7.26.2.1 OnResetTimeout upon keypress during PerformInteraction (KEYBOARD)	382

7.26.2.2 OnResetTimeout upon User's scrolling the message during ScrollableMessage	382
7.26.2.2 OnResetTimeout upon KEEP_CONTEXT soft button press during Alert	383
7.27.3 JSON Messages Examples	384
7.27 OnDriverDistraction	384
7.27.1 Description	384
7.27.1.1 Parameters	385
7.27.1.2 DriverDistractionState	385
7.27.2 Sequence Diagrams	385
7.27.2.1 OnDriverDistraction notification	385
7.27.3 JSON Messages Examples	385
7.28 OnRecordStart	386
7.29.1 Description	386
7.29.1.1 Parameters	386
7.28.2 Sequence Diagrams	386
7.28.2.1 OnRecordStart with TTS.Speak	386
7.28.2.2 OnRecordStart without TTS.Speak	387
7.28.2.3 OnRecordStart NOT sent if UI.PerformAudioPassThru is REJECTED	389
7.28.3 JSON Messages Examples	389
7.29 SetDisplayLayout	389
7.29.1 Description	389
7.29.2 Request	390
7.29.2.1 Behavior	390
7.29.2.2 Parameters	391
7.29.3 Response	391
7.29.3.1 Parameters	392
7.29.3.2 DisplayCapabilities	392
7.29.3.3 DisplayType Enumeration	393
7.29.3.4 TextField	394
7.29.3.5 TextFieldName Enumeration	394
7.29.3.6 CharacterSet	397
7.29.3.7 ImageField Structure	398
7.29.3.8 ImageFieldName Enumeration	398
7.29.3.9 FileType Enumeration	399
7.29.3.10 ImageResolution Structure	399
7.29.3.11 MediaClockFormat Enumeration	399
7.29.3.12 ImageType Enumeration	400
7.29.3.13 ScreenParams Structure	400
7.29.3.14 TouchEventCapabilities Structure	401
7.29.3.15 ButtonCapabilities	401

7.29.3.16 ButtonName	401
7.29.3.17 SoftButtonCapabilities Structure	402
7.29.3.18 PresetBankCapabilities	403
7.29.3.19 PredefinedLayout	403
7.29.4 Sequence Diagrams	407
7.29.4.1 SetDisplayLayout (successful) with preceding UI.GetCapabilities	407
7.29.4.2 SetDisplayLayout (INVALID_DATA) with preceding UI.GetCapabilities	409
7.29.5 JSON Messages Examples	410
7.29.5.1 Request.....	410
7.29.5.2 Response	410
7.29.5.3 Error message.....	413
8 Buttons Component Description.....	413
8.1 GetCapabilities	413
8.1.1 Description.....	413
8.1.2 Request	413
8.1.2.1 Behavior.....	413
8.1.3 Response.....	414
8.1.3.1 Parameters.....	415
8.1.3.2 ButtonCapabilities	415
8.1.3.3 ButtonName	415
8.1.3.4 PresetBankCapabilities	416
8.1.4 Sequence Diagrams	416
8.1.4.1 GetCapabilities on system startup	416
8.1.5 JSON Messages Examples	417
8.1.5.1 Request.....	417
8.1.5.2 Response	417
8.1.5.3 Error message.....	418
8.2 OnButtonEvent.....	418
8.2.1 Description.....	418
8.2.1.1 Parameters.....	419
8.1.3.3 ButtonName	419
8.2.1.2 ButtonEventMode.....	420
8.2.2 Sequence Diagrams	421
8.2.2.1 OnButtonEvent for CUSTOM_BUTTON being pressed and released.....	421
8.2.2.1 OnButtonEvent for hardware button being pressed and released.....	421
8.2.3 JSON Messages Examples	422
8.3 OnButtonPress	422
8.3.1 Description.....	422

8.3.1.1 Parameters.....	423
8.1.3.3 ButtonName	424
8.3.1.2 ButtonPressMode.....	425
8.3.2 Sequence Diagrams	425
8.3.2.1 OnButtonPress (SHORT) for CUSTOM_BUTTON	425
8.3.2.2 OnButtonPress (LONG) for hardware button	426
8.3.2.2 OnButtonPress for hardware button that supports SHORT press mode only	427
8.3.3 JSON Messages Examples	428
8.4 OnButtonSubscription	428
8.4.1 Description.....	428
8.4.1.1 Parameters.....	429
8.4.1.2 ButtonName	429
8.4.2 Sequence Diagrams	431
8.4.2.1 OnButtonSubscription	431
8.4.3 JSON Messages Examples	431
9 VR Component Description.....	432
9.1 IsReady	432
9.1.1 Description.....	432
9.1.2 Request	432
7.1.2.1 Behavior	432
9.1.3 Response.....	432
9.1.3.1 Parameters.....	433
9.1.4 Sequence Diagrams	433
9.1.4.1 VR.IsReady	433
9.1.5 JSON Messages Examples	433
9.1.5.1 Request.....	433
9.1.5.2 Response	434
9.1.5.3 Error message.....	434
9.2 GetCapabilities	434
9.2.1 Description.....	434
9.2.2 Request	434
9.2.2.1 Behavior	434
9.2.3 Response.....	435
9.2.3.1 Parameters.....	435
9.2.3.2 VrCapabilities	435
9.2.4 Sequence Diagrams	435
9.2.4.1 VR.GetCapabilities	435
9.2.5 JSON Messages Examples	436
9.2.5.1 Request.....	436

9.2.5.2 Response	436
9.2.5.3 Error message.....	437
9.3 GetSupportedLanguages.....	437
9.3.1 Description.....	437
9.3.2 Request	437
7.1.2.1 Behavior	437
9.3.3 Response.....	437
9.3.3.1 Parameters.....	438
9.3.3.2 Language Enumeration.....	438
9.3.4 Sequence Diagrams	439
9.3.4.1 VR.GetSupportedLanguages	439
9.3.5 JSON Messages Examples	439
9.3.5.1 Request.....	439
9.3.5.2 Response	439
9.3.5.3 Error message.....	440
9.4 AddCommand	440
9.4.1 Description.....	440
9.4.2 Request	440
9.4.2.1 Behavior	440
9.4.2.2 Parameters.....	441
9.4.2.3 VR CommandType.....	442
9.4.3 Response.....	442
9.4.4 Sequence Diagrams	443
9.4.4.1 VR.AddCommand	443
9.4.4.2 UI.AddCommand returns SUCCESS, VR portion failed.....	444
9.4.4.3 UI.AddCommand returns SUCCESS, VR portion no response	444
9.4.4.4 UI.AddCommand fails, VR portion SUCCESS	445
9.4.4.5 UI.AddCommand no response, VR portion SUCCESS	445
9.4.5 JSON Messages Examples	445
9.4.5.1 Request.....	445
9.4.5.2 Response	446
9.4.5.3 Error message.....	446
9.5 DeleteCommand	446
9.5.1 Description.....	446
9.5.2 Request	447
9.5.2.1 Behavior	447
9.5.2.2 Parameters.....	447
9.5.3 Response.....	447
9.5.4 Sequence Diagrams	448

9.1.4.1 VR.DeleteCommand	448
9.5.5 JSON Messages Examples	449
9.5.5.1 Request.....	449
9.5.5.2 Response	449
9.5.5.3 Error message.....	450
9.6 ChangeRegistration	450
9.6.1 Description.....	450
9.6.2 Request	450
9.6.2.1 Behavior	450
9.6.2.2 Parameters.....	450
9.6.2.2 Language	451
9.6.3 Response.....	452
9.6.4 Sequence Diagrams	452
9.6.4.1 VR.ChangeRegistration	452
9.6.5 JSON Messages Examples	453
9.6.5.1 Request.....	453
9.6.5.2 Response	454
9.6.5.3 Error message.....	454
9.7 GetLanguage	454
9.7.1 Description.....	454
9.7.2 Request	454
9.7.2.1 Behavior	454
9.7.3 Response.....	454
9.7.3.1 Parameters.....	455
9.7.3.2 Language	455
9.7.4 Sequence Diagrams	456
9.7.4.1 VR.GetLanguage	456
9.7.5 JSON Messages Examples	456
9.7.5.1 Request.....	456
9.7.5.2 Response	457
9.7.5.3 Error message.....	457
9.8 Started.....	457
9.8.1 Description.....	457
9.8.2 Sequence Diagrams	458
9.8.2.1 VR.Started on PTT button press	458
9.8.3 JSON Messages Examples	458
9.9 Stopped.....	459
9.9.1 Description.....	459
9.9.2 Sequence Diagrams	460
9.8.2.1 VR.Stopped on VR session ending	460

9.9.3 JSON Messages Examples	460
9.10 OnCommand.....	461
9.10.1 Description.....	461
9.10.1.1 Parameters.....	461
9.10.2 Sequence Diagrams	462
9.10.2.1 VR.OnCommand.....	462
9.10.3 JSON Messages Examples	462
9.11 OnLanguageChange.....	462
9.11.1 Description.....	462
9.11.1.1 Parameters.....	463
9.11.1.2 Language	463
9.11.2 Sequence Diagrams	464
9.11.2.1 VR.OnLanguageChange.....	464
9.11.3 JSON Messages Examples	465
9.12 VR.PerformInteraction.....	466
9.12.1 Description.....	466
9.12.2 Request	466
9.12.2.1 Behavior	466
9.12.2.2 Parameters.....	469
9.12.3 Response.....	469
9.12.4 Parameters.....	469
9.12.4 Sequence Diagrams	470
9.12.4.1 VR.PerfromInteraction in 'VR only' mode successfully completed	470
9.12.4.2 VR.PerfromInteraction in 'Manual only' mode successfully completed	470
9.12.4.3 VR.PerfromInteraction in 'Both' mode timed out.....	471
9.12.5 JSON Messages Examples	472
9.12.5.1 Request.....	472
9.12.5.2 Response	473
9.12.5.3 Error message.....	473
10 TTS Component Description	474
10.1 IsReady	474
10.1.1 Description.....	474
10.1.2 Request	474
10.1.2.1 Behavior	474
10.1.3 Response.....	474
10.1.3.1 Parameters.....	474
10.1.4 Sequence Diagrams	475
10.1.4.1 TTS.IsReady and preceding OnReady	475
10.1.5 JSON Messages Examples	475

10.1.5.1 Request	475
10.1.5.2 Response	475
10.1.5.3 Error message.....	475
10.2 GetCapabilities	476
10.2.1 Description.....	476
10.2.2 Request	476
7.1.2.1 Behavior.....	476
10.2.3 Response.....	476
10.2.3.1 Parameters.....	477
10.2.3.2 SpeechCapabilities	477
10.2.3.3 PrerecordedSpeech	477
10.2.4 Sequence Diagrams	477
10.1.4.1 TTS.GetCapabilities	477
10.2.5 JSON Messages Examples	478
10.2.5.1 Request.....	478
10.2.5.2 Response	478
10.2.5.3 Error message.....	478
10.3 GetSupportedLanguages	479
10.3.1 Description.....	479
10.3.2 Request	479
7.1.2.1 Behavior.....	479
10.3.3 Response.....	479
10.3.3.1 Parameters.....	480
10.3.3.2 Language	480
10.3.4 Sequence Diagrams	481
10.3.4.1 TTS.GetSupportedLanguages	481
10.3.5 JSON Messages Examples	481
10.3.5.1 Request.....	481
10.3.5.2 Response	481
10.3.5.3 Error message.....	482
10.4 Speak	482
10.4.1 Description.....	482
10.4.2 Request	482
10.4.2.1 Behavior	482
10.4.2.2 Parameters.....	483
10.4.2.3 TTSShunk	483
10.4.2.4 SpeechCapabilities	483
10.4.2.5 MethodName.....	484
10.4.3 Response.....	484
10.4.4 Sequence Diagrams	485
10.4.4.1 Speak	485

10.4.5 JSON Messages Examples	485
10.4.5.1 Request.....	485
10.4.5.2 Response	485
10.4.5.3 Error message.....	486
10.5 StopSpeaking.....	486
10.5.1 Description.....	486
10.5.2 Request	486
10.5.2.1 Behavior	486
10.5.3 Response.....	486
10.5.4 Sequence Diagrams	487
10.5.4.1 StopSpeaking	487
10.5.5 JSON Messages Examples	487
10.5.5.1 Request.....	487
10.5.5.2 Response	487
10.5.5.3 Error message.....	488
10.6 ChangeRegistration	488
10.6.1 Description.....	488
10.6.2 Request	488
10.6.2.1 Behavior	488
10.6.2.2 Parameters.....	489
10.6.2.3 Language	489
10.6.3 Response.....	490
10.6.4 Sequence Diagrams	490
10.6.4.1 TTS.ChangeRegistration after OnAppRegistered.....	490
10.6.5 JSON Messages Examples	491
10.6.5.1 Request.....	491
10.6.5.2 Response	492
10.6.5.3 Error message.....	492
10.7 GetLanguage	492
10.7.1 Description.....	492
10.7.2 Request	492
10.7.2.1 Behavior	492
10.7.3 Response.....	492
10.7.3.1 Parameters.....	493
10.6.2.3 Language	493
10.7.4 Sequence Diagrams	494
10.7.4.1 TTS.GetLanguage.....	494
10.7.5 JSON Messages Examples	494
10.7.5.1 Request.....	494
10.7.5.2 Response	494

10.7.5.3 Error message.....	495
10.8 SetGlobalProperties	495
10.8.1 Description	495
10.8.2 Request	495
10.8.2.1 Behavior	495
10.8.2.2 Parameters.....	496
10.8.3 Response.....	496
10.8.4 Sequence Diagrams	497
10.8.4.1 TTS.SetGlobalProperties and corresponding HMI processing	497
10.8.5 JSON Messages Examples	499
10.8.5.1 Request.....	499
10.8.5.2 Response	499
10.8.5.3 Error message.....	499
10.10 OnLanguageChange.....	499
10.10.1 Description.....	499
10.10.1.1 Parameters.....	500
10.10.1.2 Language	500
10.10.2 Sequence Diagrams	501
10.10.2.1 TTS.OnLanguageChange	501
10.10.3 JSON Messages Examples	502
10.11 Started.....	502
10.11.1 Description.....	502
10.11.2 Request	502
10.11.2.1 Parameters.....	502
10.11.3 Response.....	502
10.11.3 Sequence Diagrams	503
10.11.3.1 TTS.Started upon TTS.Speak request from SDL	503
10.11.3.2 TTS.Started during PerfromInteraction	503
10.11.4 JSON Messages Examples	504
10.11.4.1 Request.....	504
10.11.4.2 Response	504
10.11.4.3 Error message.....	505
10.12 Stopped.....	505
10.12.1 Description.....	505
10.12.2 Request	505
10.12.2.1 Parameters.....	505
10.12.3 Response.....	505
10.12.4 Sequence Diagrams	506

10.12.4.1 TTS.Stopped after TTS.StopSpeaking from SDL	506
10.12.5 JSON Messages Examples	506
10.12.5.1 Request	506
10.12.5.2 Response	507
10.12.5.3 Error message	507
10.13 OnResetTimeout	507
10.13.1 Description	507
10.13.1.1 Parameters	507
10.13.2 Sequence Diagrams	508
10.13.2.1 OnResetTimeout for TTS.Speak SUCCESS	508
10.13.2.2 OnResetTimeout for TTS.Speak GENERIC ERROR	509
10.13.3 JSON Messages Examples	509
11 VehicleInfo Component Description	509
11.1 IsReady	510
11.1.1 Description	510
11.1.2 Request	510
11.1.2.1 Behavior	510
11.1.3 Response	510
11.1.3.1 Parameters	510
11.1.4 Sequence Diagrams	511
11.1.4.1 VehicleInfo.IsReady and preceding OnReady	511
11.1.5 JSON Messages Examples	511
11.1.5.1 Request	511
11.1.5.2 Response	511
11.1.5.3 Error message	511
11.2 GetVehicleType	512
11.2.1 Description	512
11.2.2 Request	512
11.2.2.1 Behavior	512
11.2.3 Response	512
11.2.3.1 Parameters	513
11.2.3.2 VehicleType	513
11.2.4 Sequence Diagrams	513
11.2.4.1 GetVehicleType	513
11.2.5 JSON Messages Examples	514
11.2.5.1 Request	514
11.2.5.2 Response	514
11.2.5.3 Error message	514
11.3 ReadDID	514

11.3.1 Description	514
11.3.2 Request	515
11.3.2.1 Behavior	515
11.3.2.1 Parameters.....	515
11.3.3 Response.....	515
11.3.3.1 Parameters.....	516
11.3.3.2 DIDResult.....	516
11.3.3.3 VehicleDataresultCode	517
11.3.4 Sequence Diagrams	517
11.3.4.1 ReadDID general processing	517
11.3.4.2 ReadDID with expanded didResult in response.....	517
11.3.5 JSON Messages Examples	518
11.3.5.1 Request.....	518
11.3.5.2 Response	518
11.3.5.3 Error message.....	519
11.4 GetDTCs	519
11.4.1 Description	519
11.4.2 Request	519
11.4.2.1 Behavior	519
11.4.2.1 Parameters.....	520
11.4.3 Response.....	520
11.4.3.1 Parameters.....	521
11.4.4 Sequence Diagrams	521
11.4.4.1 GetDTCs	521
11.4.5 JSON Messages Examples	521
11.4.5.1 Request.....	521
11.4.5.2 Response	522
11.4.5.3 Error message.....	522
11.5 DiagnosticMessage	522
11.5.1 Description	522
11.5.2 Request	522
11.5.2.1 Behavior	522
11.5.2.1 Parameters.....	523
11.5.3 Response.....	523
11.5.3.1 Parameters.....	524
11.5.4 Sequence Diagrams	524
11.5.4.1 DiagnosticMessage	524
11.5.5 JSON Messages Examples	524
11.5.5.1 Request.....	524
11.5.5.2 Response	525
11.5.5.3 Error message.....	525

11.6 SubscribeVehicleData.....	525
11.6.1 Description.....	525
11.6.2 Request	525
11.6.2.1 Behavior	525
11.6.2.2 Parameters.....	526
11.6.3 Response.....	527
11.6.3.1 Parameters.....	528
11.6.3.2 VehicleDataResult.....	529
11.6.3.3 VehicleDataType Enumeration	529
FORD specific VehicleDataType Enumeration (extension of 11.6.3.3 table).....	530
11.6.3.4 VehicleDataresultCode Enumeration.....	531
11.6.4 Sequence Diagrams	532
11.6.4.1 SubscribeVehicleData	532
11.6.5 JSON Messages Examples	532
11.6.5.1 Request.....	532
11.6.5.2 Response	533
11.6.5.3 Error message.....	535
11.7 UnSubscribeVehicleData	535
11.7.1 Description.....	535
11.7.2 Request	535
11.7.2.1 Behavior	535
11.7.2.2 Parameters.....	536
11.7.3 Response.....	537
11.7.3.1 Parameters.....	538
11.7.3.2 VehicleDataResult.....	539
11.7.3. VehicleDataType Enumeration	539
FORD specific VehicleDataType Enumeration (extension of 11.7.3.3 table).....	540
11.7.3. VehicleDataresultCode Enumeration.....	541
11.7.4 Sequence Diagrams	542
11.7.4.1 UnSubscribeVehicleData	542
11.7.4.2 UnSubscribeVehicleData unexpected disconnect	544
11.7.5 JSON Messages Examples	544
11.7.5.1 Request.....	544
11.7.5.2 Response	545
11.7.5.3 Error message.....	547
11.8 GetVehicleData.....	547
11.8.1 Description.....	547
11.8.2 Request	547
11.8.2.1 Behavior	547

11.8.2.2 Parameters.....	548
11.8.3 Response.....	549
11.8.3.1 Parameters.....	549
11.8.3.2 GPSData	551
11.8.3.3 CompassDirection.....	552
11.8.3.4 Dimension	552
11.8.3.5 ComponentVolumeStatus	552
11.8.3.6 PRNDL	552
11.8.3.7 TireStatus	553
11.8.3.8 WarningLightStatus.....	553
11.8.3.9 SingleTireStatus	553
11.8.3.10 BeltStatus	553
11.8.3.13 VehicleDataEventStatus	554
11.8.3.15 EmergencyEventType	554
11.8.4 Sequence Diagrams	555
11.8.4.1 GetVehicleData	555
11.8.5 JSON Messages Examples	555
11.8.5.1 Request.....	555
11.8.5.2 Response	556
11.8.5.3 Error message.....	557
11.9 OnVehicleData	558
11.9.1 Description.....	558
11.9.1.1 Parameters.....	558
11.9.2.6 VehicleDataEventStatus	560
11.9.2 Sequence Diagrams	561
11.9.2.1 OnVehicleData	561
11.9.3 JSON Messages Examples	561
12 Navigation Component Description	562
12.1 IsReady	562
12.1.1 Description.....	562
12.1.2 Request	562
12.1.2.1 Behavior	562
12.1.3 Response.....	562
12.1.3.1 Parameters.....	563
12.1.4 Sequence Diagrams	563
12.1.4.1 Navigation.IsReady and preceding OnReady	563
12.1.5 JSON Messages Examples	563
12.1.5.1 Request.....	563
12.1.5.2 Response	563
12.1.5.3 Error message.....	563
12.2 AlertManeuver.....	564

12.2.1 Description	564
12.2.2 Request	564
12.2.2.1 Behavior	564
12.1.2.1 Parameters.....	565
12.2.3 Response.....	565
12.2.4 Sequence Diagrams	567
12.2.4.1 AlertManeuver Default success path	567
12.2.4.2 AlertManeuver from BACKGROUND success path	568
12.2.4.3 AlertManeuver ABORTED	569
12.2.4.4 AlertManeuver REJECTED.....	570
12.2.5 JSON Messages Examples	570
12.2.5.1 Request Navi.AlertManeuver	570
12.2.5.2 Response Navi.AlertManeuver	571
12.2.5.3 Request TTS.Speak for AlertManeuver	571
12.2.5.4 Response TTS.Speak for AlertManeuver	571
12.2.5.5 Error message.....	571
12.3 ShowConstantTBT	572
12.3.1 Description.....	572
12.3.2 Request	572
12.3.2.1 Parameters.....	572
12.3.3 Response.....	573
12.3.4 Sequence Diagrams	574
12.3.4.1 ShowConstantTBT	574
12.3.5 JSON Messages Examples	575
12.3.5.1 Request.....	575
12.3.5.2 Response	575
12.3.5.3 Error message.....	576
12.4 UpdateTurnList.....	576
12.4.1 Description	576
12.4.2 Request	576
12.4.2.1 Behavior	576
12.4.2.1 Parameters.....	576
12.4.2.2 Turn.....	577
12.4.3 Response.....	577
12.4.4 Sequence Diagrams	578
12.4.4.1 UpdateTurnList.....	578
12.4.5 JSON Messages Examples	578
12.4.5.1 Request.....	578
12.4.5.2 Response	580
12.4.5.3 Error message.....	580

12.5 StartStream	580
12.5.1 Description.....	580
12.5.2 Request	581
12.5.2.1 Parameters.....	581
12.5.3 Response.....	581
12.5.3.1 Parameters.....	581
12.5.4 Sequence Diagrams	582
12.5.4.1 StartStream	582
12.5.5 JSON Messages Examples	583
12.5.5.1 Request.....	583
12.5.5.2 Response	583
12.5.5.3 Error message.....	583
12.6 StopStream	583
12.6.1 Description.....	583
12.6.2 Request	584
12.6.2.1 Parameters.....	584
12.6.3 Response.....	584
12.6.3.1 Parameters.....	584
12.6.4 Sequence Diagrams	584
12.6.2.1 StopStream	584
12.6.5 JSON Messages Examples	585
12.6.5.1 Request.....	585
12.6.5.2 Response	586
12.6.5.3 Error message.....	586
12.7 StartAudioStream.....	586
12.7.1 Description.....	586
12.7.2 Request	587
12.7.1.1 Parameters.....	587
12.7.3 Response.....	587
12.7.3.1 Parameters.....	587
12.7.4 Sequence Diagrams	587
12.5.4.1 StartAudioStream/StopAudioStream/OnAudioDataStream.....	587
12.7.5 JSON Messages Examples	588
12.7.5.1 Request.....	588
12.7.5.2 Response	589
12.7.5.3 Error message.....	589
12.8 StopAudioStream	589
12.8.1 Description	589
12.8.2 Request	590

12.8.2.1 Parameters.....	590
12.8.3 Response.....	590
12.8.3.1 Parameters.....	590
12.8.4 Sequence Diagrams	591
12.8.4.1 StopAudioStream	591
12.8.5 JSON Messages Examples	592
12.8.5.1 Request.....	592
12.8.5.2 Response	592
12.8.5.3 Error message.....	592
12.9 OnTBTClientState	592
12.9.1 Description.....	592
12.9.1.1 Parameters.....	593
12.9.1.2 TBTState	593
12.9.2 Sequence Diagrams	593
12.9.2.1 OnTBTClientState ROUTE_UPDATE_REQUEST	593
12.9.3 JSON Messages Examples	594
12.10 SendLocation	594
12.10.1 Description.....	594
12.10.2 Request	594
12.10.2.1 Behavior	594
12.10.2.2 Parameters.....	595
12.10.3 Response.....	595
12.10.4 Sequence Diagrams	597
12.10.4.1 SendLocation Success scenario	597
12.10.4.2 SendLocation Failure scenario WARNINGS	597
12.10.4.3 SendLocation Failure scenario REJECTED.....	598
12.10.5 JSON Messages Examples	598
12.10.5.1 Request.....	598
12.10.5.2 Response	598
12.10.5.3 Error message.....	599
12.11 OnAudioDataStreaming	599
12.11.1 Description.....	599
12.11.1.1 Parameters.....	599
12.11.2 Sequence Diagrams	599
12.11.2.1 OnAudioDataStreaming SDL<->HMI	599
12.11.2.1 OnAudioDataStreaming full diagram App<->HMI.....	601
12.11.3 JSON Messages Examples	601
12.12 OnVideoDataStreaming	602
12.12.1 Description.....	602

12.12.1.1 Parameters.....	602
12.12.2 Sequence Diagrams	602
12.12.2.1 OnVideoDataStreaming	602
12.12.3 JSON Messages Examples.....	603
13 SDL Component Description	604
13.1 SDL.ActivateApp	604
13.1.1 Description.....	604
13.1.2 Request	604
13.1.2.1 Behavior	604
13.1.2.2 Parameters.....	606
13.1.3 Response.....	606
13.1.3.1 Parameters.....	607
13.1.3.2 AppPriority.....	607
13.1.3.3 DeviceInfo Structure.....	608
13.1.3.4 TransportType Enumeration	608
13.1.4 Sequence Diagrams	609
13.1.4.1 SDL.ActivateApp for the application registered and this application was reduced in permissions by the latest PT Update.	609
13.1.4.2 SDL.ActivateApp for the application registered and this application was unauthorized by the latest PT Update.....	610
13.1.4.3 SDL.ActivateApp in App Launching and Querying sequence	611
13.1.5 JSON Messages Examples	613
13.1.5.1 Request.....	613
13.1.5.2 Response	613
13.1.5.3 Error message.....	613
13.2 GetListOfPermissions	613
13.2.1 Description.....	613
13.2.2 Request	614
13.2.2.1 Behavior	614
13.2.1.1 Parameters.....	615
13.2.3 Response.....	615
13.2.3.1 Parameters.....	615
13.2.3.2 PermissionItem	615
13.2.4 Sequence Diagrams	616
13.2.4.1 GetListOfPermissions	616
13.2.4.2 GetListOfPermissions without appId	617
13.2.5 JSON Messages Examples	617
13.2.5.1 Request.....	617
13.2.5.2 Response	617

13.2.5.3 Error message.....	618
13.3 GetStatusUpdate.....	618
13.3.1 Description	618
13.3.2 Request	618
13.3.2.1 Behavior	618
13.3.3 Response.....	619
13.3.3.1 Parameters.....	619
See 13.3.3.2 UpdateResult	619
13.3.3.2 UpdateResult	619
13.3.4 Sequence Diagrams	619
13.3.4.1 GetStatusUpdate	619
13.3.5 JSON Messages Examples	620
13.3.5.1 Request.....	620
13.3.5.2 Response	620
13.3.5.3 Error message.....	620
13.4 GetUserFriendlyMessage	621
13.4.1 Description.....	621
13.4.2 Request	621
13.4.2.1 Behavior	621
13.4.2.2 Parameters.....	621
13.4.2.3 MessageCodes	621
13.4.3 Response.....	622
13.4.3.1 Parameters.....	622
13.4.3.2 UserFriendlyMessage	622
13.4.4 Sequence Diagrams	623
13.4.4.1 GetUserFriendlyMessage for DeviceConsent.....	623
13.4.5 JSON Messages Examples	624
13.4.5.1 Request.....	624
13.4.5.2 Response	625
13.4.5.3 Error message.....	625
13.5 OnAppPermissionChanged.....	625
13.9.1 Description	625
13.5.1.1 Parameters.....	626
13.5.1.3 AppPriority.....	626
13.5.1.3 RequestType Enumeration	627
13.5.2 Sequence Diagrams	627
13.5.2.1 OnAppPermissionChanged with appPermissionsConsentNeeded:true @TODO to update	627
13.5.3 JSON Messages Examples	630
13.6 OnStatusUpdate.....	630
13.6.1 Description	630

13.6.1.1 Parameters.....	630
13.6.1.2 UpdateResult	630
13.6.2 Sequence Diagrams	631
13.6.2.1 OnStatusUpdate.....	631
13.6.3 JSON Messages Examples	632
14 Common Component Description	632
14.1 Enumerations.....	632
14.1.1 Result.....	632
14.1.2 AppHMIType	635
14.1.3 DeactivateReason	635
14.1.4 ApplicationsCloseReason.....	636
14.1.5 ClockUpdateMode	636
14.1.6 SystemContext	637
14.1.7 DisplayType	637
14.1.8 MediaClockFormat.....	638
14.1.9 HmiZoneCapabilities.....	640
14.1.10 SpeechCapabilities	640
14.1.11 VrCapabilities.....	640
14.1.12 DriverDistractionState	641
14.1.11 SoftButtonType	641
14.1.12 SystemAction.....	641
14.1.13 TextAlignement.....	642
14.1.14 Language	643
14.1.15 TextFieldName	644
14.1.17 ImageFieldName	649
14.1.18 ImageType	650
14.1.17 ButtonName	650
14.1.18 ButtonEventMode	651
14.1.19 ButtonPressMode	652
14.1.20 LayoutMode	652
14.1.21 TouchEvent.....	653
14.1.22 SamplingRate	653
14.1.23 BitsPerSample	654
14.1.24 AudioType.....	654
14.1.25 KeyboardLayout.....	654
14.1.26 KeyboardEvent	655
14.1.27 KeypressMode	655
14.1.28 AmbientLightStatus.....	656
14.1.29 ECallConfirmationStatus.....	656

14.1.30	VehicleDataNotificationStatus	657
14.1.31	EmergencyEventType	657
14.1.32	FuelCutoffStatus	658
14.1.33	PowerModeQualificationStatus.....	658
14.1.34	CarModeStatus.....	659
14.1.35	PowerModeStatus	659
14.1.36	ComponentVolumeStatus.....	660
14.1.37	PRNDL.....	660
14.1.38	WarningLightStatus	661
14.1.39	VehicleDataEventStatus	661
14.1.40	IgnitionStableStatus.....	662
14.1.41	IgnitionStatus	662
14.1.42	DeviceLevelStatus	663
14.1.43	Primary AudioSource	663
14.1.45	WiperStatus	664
14.1.46	CompassDirection	664
14.1.47	Dimension	665
14.1.48	VehicleDataResultCode.....	665
14.1.49	TBTState.....	666
14.1.50	MethodName	666
14.1.51	EmergencyState	667
14.1.52	TransportType	667
14.2	Structures.....	667
14.2.1	HMIApplication.....	667
14.2.2	DeviceInfo.....	669
14.2.3	MenuParams	670
14.2.4	Choice.....	670
14.2.5	TimeFormat	672
14.2.6	VrHelpItem.....	673
14.2.7	DisplayCapabilities	673
14.2.8	TouchEventCapabilities	674
14.2.9	ImageResolution	675
14.2.10	ScreenParams	675
14.2.11	ImageField	676
14.2.12	TextFieldStruct.....	676
14.2.13	SoftButtonCapabilities	677
14.2.14	SoftButton	678
14.2.15	Image	679
14.2.16	ButtonCapabilities	679

14.2.17	PresetBankCapabilities.....	680
14.2.18	AudioPassThruCapabilities.....	681
14.2.19	TouchCoord	681
14.2.20	TouchEvent.....	682
14.2.21	KeyboardProperties	682
14.2.22	TTSChunk.....	683
14.2.23	VehicleType	683
14.2.24	DIDResult	684
14.2.25	GPSData.....	684
14.2.26	TireStatus	686
14.2.27	SingleTireStatus	686
14.2.28	BeltStatus	686
14.2.29	BodyInformation.....	687
14.2.30	DeviceStatus.....	688
14.2.31	HeadLampStatus	689
14.2.32	Turn	689
14.2.33	HMICapabilities.....	690
15	Other	690
15.1	SDL's configuration file structure (ini-file).....	690
15.1.1	Description.....	690
15.1.2	Structure	690
15.1.2.1	HMI.....	690
15.1.2.2	MAIN	691
15.1.2.4	MEDIA MANAGER.....	693
15.1.2.5	GLOBAL PROPERTIES.....	695
15.1.2.6	FILESYSTEM RESTRICTIONS	696
15.1.2.6	VR COMMANDS	697
15.1.2.7	ApplInfo.....	697
15.1.2.8	Policy.....	697
15.1.2.9	TransportManager.....	698
15.1.2.10	ProtocolHandler	698
15.1.2.11	Security Manager	699
15.1.2.12	ApplicationManager	700
15.1.2.13	SDL4	701
15.1.2.14	Resumption	701
15.2	HMI<->SDL fake parameters processing.....	702
15.2.1	Description.....	702
15.2.2	Sequence diagrams.....	703
15.2.2.1	Fake parameters cut off and the valid data left.....	703

15.2.2.2 Fake parameters cut off and invalid data left (no mobile API relationship)	703
15.2.2.3 Fake parameters cut off and invalid data left related to mobile API	704
15.3 HMI Capabilities JSON file	704
15.3.1 Description	704
15.3.2 Sections	704
15.3.2.1 UI.....	704
15.3.2.2 VR	706
15.3.2.3 TTS	706
15.3.2.4 Buttons	706
15.3.2.5 VehicleInfo	707
15.3.2.6 SyncMessageVersion	707
15.3.3 Example.....	708
References.....	716
Change History	717
1) Section #7.12.2.2 SetGlobalProperties Parameters	717
- added reference for keyboardProperties to hmi_capabilities.json section.....	717
2) Section #8.1 Buttons.GetCapabilities	717
- added SDL note about hmi_capabilities.json.....	717
3) Section #9.2 VR.GetCapabilities.....	717
- updated SDL note about hmi_capabilities.json.....	717
4) Section #10.2 TTS.GetCapabilities.....	717
- updated SDL note about hmi_capabilities.json.....	717
5) Section #15.1.2.1 smartDeviceLink.ini HMI	717
- make reference to appropriate chapters in Desctiption	717
- added LinkToWebHMI parameter.....	717
6) Section #15.1.2.2 smartDeviceLink.ini MAIN.....	717
- added SDLVersion parameter	718
- added AppResourceFolder parameter.....	718
- updated AppStorageFolder parameter description	718
Section #15.1.2.4 smartDeviceLink.ini MEDIA MANAGER	718
- updated description for NamedVideoPipePath and NamedAudioPipePath.....	718

Introduction

This document provides the information for integrating the system of SmartDeviceLink (SDL) with the vehicle Head Unit (HU).

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

With the help of SDL the applications running on the mobile devices with:

- Operating system of Android / iOS / BlackBerry
- Connection over Bluetooth / USB / WiFi

might be presented to vehicle HMI.

The given guideline describes:

- The types of transports supported by SDL for communication with HMI
- How to connect over one of the supported transports
- The types of messages and the message formats used for communication
- The API that needs to be supported by vehicle HMI.

The document also provides:

- The drawings showing the exemplary display layout for illustrating the expected HMI behavior.
- The sequence diagrams for showing the sequence of messages to be received/responded by both HMI and SDL.
- The examples for each function call in different message formats corresponding to the transports currently supported.

Abbreviations and Definitions

Abbreviations used in this document are collected in the table below.

Abbreviation	Expansion
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 Precision
RPC	Remote Procedure Call
SDE	Software Development Environment
SDL	SmartDeviceLink
SEE	Software Engineering Environment
TTS	Text To Speech
VDOP	Vertical Dilution Of Precision
VR	Voice Recognition
UTC	Universal Time Coordinate

The list of terms used in the document and their definitions is provided in the table below.

Term	Definition
Data Identifier (DID)	Addressed memory locations of the ECU.
Diagnostic Trouble Code	In the automotive industry, codes that are prescribed by SAE standards to help track problems in a vehicle detected by its on-board computer

JSON	<ul style="list-style-type: none"> - Is a lightweight data-interchange format. It is easy for humans to read and write. It is easy for machines to parse and generate. - Is a text format that is completely language independent but uses conventions that are familiar to programmers of the C-family of languages.
GPS	GPS – The Global Positioning System – is a space-based satellite navigation system that provides location (using not less than four satellites simultaneously) and time information to anyone who exploits GPS receiver. When installed in a car, a GPS unit can provide useful information about the car's position and the best travel routes to a given destination.
E911 override	Occurs in case of a car accident – when the airbag deploys or, in certain vehicles, the emergency fuel pump shutoff is activated. The vehicle system (in case of relevant version) should react in appropriate way [3] when this override happens.
VR	<p>Voice recognition is the technology by which sounds, words or phrases spoken by humans are converted into electrical signals and these signals are transformed into coding patterns to which meaning has been assigned.</p> <p>Relating to automotive, voice recognition aims to enable drivers to control entertainment and navigation systems simply by using their voices.</p>
Command	This is an option that a user can select either via voice recognition (VR) or through the on-screen menu during a user-initiated interaction.
Phoneme	A <i>phoneme</i> is the basic unit of sound in a language.

I. SDL Programmer's Guide

1 Overview

2 Architecture

3 Transport Manager

3.1 General description

3.1.1 Transport layer features

Several definitions:

- 1) '*Connection establishing*' with the application or device:
 - Means that transport layer creates a physical interconnection for sending and receiving 'handshake' data.
 - Does not mean that this application or device shall be marked as "connected" for the user.
 - May have two options:
 - One-time connection: when the data related to a device is cleared up after disconnection
 - Persistent connection: when the information about the device is stored even in the power off mode.
- 2) '*Connection closing*':
 - Means that connection is not terminated until 'goodbye' data transmission is finished (all accumulated data is sent to / received from the application).
 - Physical disconnection happens when there is no more data left for transmission.

Table #1: List of features provided by transport layer

N	Feature
1.	Search for applications on devices using all available transports
a)	Notify with the list of applications found on device
b)	Notify on search finished in adapter
c)	Notify on search done
d)	Inform for which transport the connection has failed
2.	Establish a connection with the user-selected application for specific device using specific transport
a)	Notify on connection successful setup
b)	Notify on connection failure
3.	Establish a connection with the user-selected application for specific device on all transports supported by the application
a)	Notify on connection successful setup
b)	Inform for which transport the connection has failed
4.	Establish a connection with all applications for the user-selected device (using specific or all supported transports)
a)	Notify on connection successful setup
b)	Inform for which application (and transport) the connection has failed
5.	Establish a connection with all applications on all user-selected devices (using specific or all supported transports)
a)	Notify on connection successful setup
b)	Inform for which application, device and/or transport

	the connection has failed
6.	Establish a connection with the application that requests an access to the system by all available transports
a)	Notify on connection request from the application on unknown device
7.	Remember the device (device is marked in a special way and is kept in the system until the user asks to clear all data connected with it)
8.	'Forget' the device (removing the mark from the device and removing the device from internal storage)
9.	Enable/Disable SDL visibility for devices (similar to BT visible/invisible)
10.	Close connection for specific application on specific device of specific transport
a)	Notify on successful disconnection
b)	Notify on disconnection failure
11.	Close connection for specific application on specific device of all transports
a)	Notify on successful disconnection
b)	Notify on disconnection failure
12.	Close connection for specific device of specific transport
a)	Notify on successful disconnection
b)	Notify on disconnection failure
13.	Close connection for specific device of all available transports
a)	Notify on successful disconnection
b)	Notify on disconnection failure
14.	Close connection for all devices of all transports
a)	Notify on successful disconnection
b)	Notify on disconnection failure
15.	Transmit data to a connected device
a)	Transmit data with the use of priority
b)	Notify on successful data transmission
c)	Inform which part of transmitted data stream has failed
16.	Receive data from a connected device
a)	Process received data with the use of priority
b)	Notify on successful data receipt
17.	Reconnect the unexpectedly disconnected application (device)
a)	Accumulate data destined for unexpectedly disconnected device
b)	Transmit all accumulated data in correct order if the application has reconnected
c)	Inform that the accumulated data has not been sent if there is no connection established after the timeout.
18.	Restore previous state
a)	Save the internal information about the connected devices in persistent memory before the shutdown
b)	Load the information about connected devices from persistent memory
19.	Reconnect applications configured to auto reconnect on power on
20.	Automatically select transport according to

	priority for multi-transport applications (if appropriately configured)
21.	Automatically switch the data flow for unexpectedly disconnected transport to another transport for multi-transport applications (if appropriately configured)
22.	Provide current state information
a)	List the connected devices providing the list of applications for each device
b)	List the existing connections providing the information on their status
c)	List the available transports

3.1.2 Transport Level Structure

The *Figure 1* demonstrates the structure of the Transport Level:

- Transport Manager has no limitations on transports number.
- Each transport has no limitations on devices number.
- Any device may be connected through unlimited number of transports. In this case each connection established between the application on a device and the Transport Manager has the unique ID.

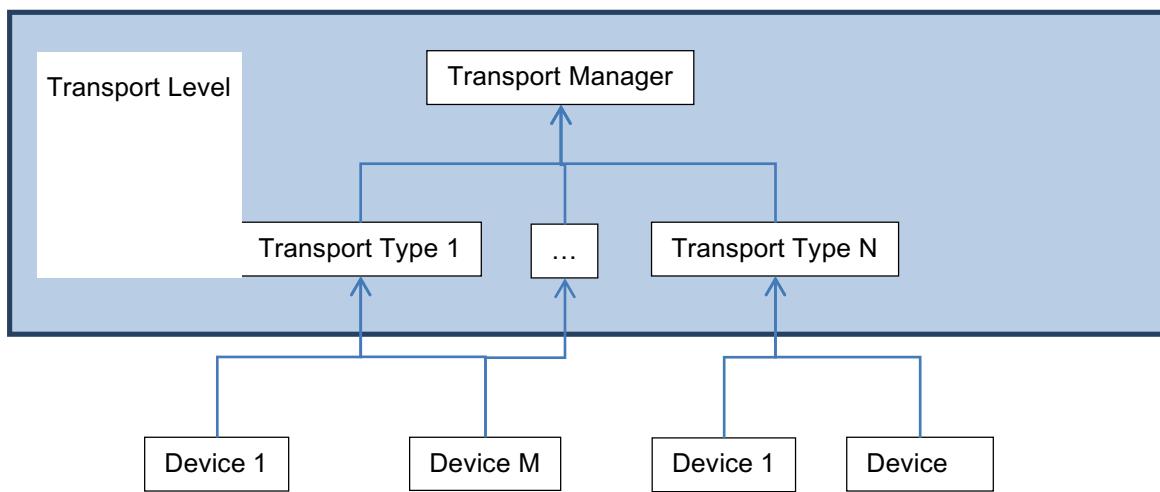
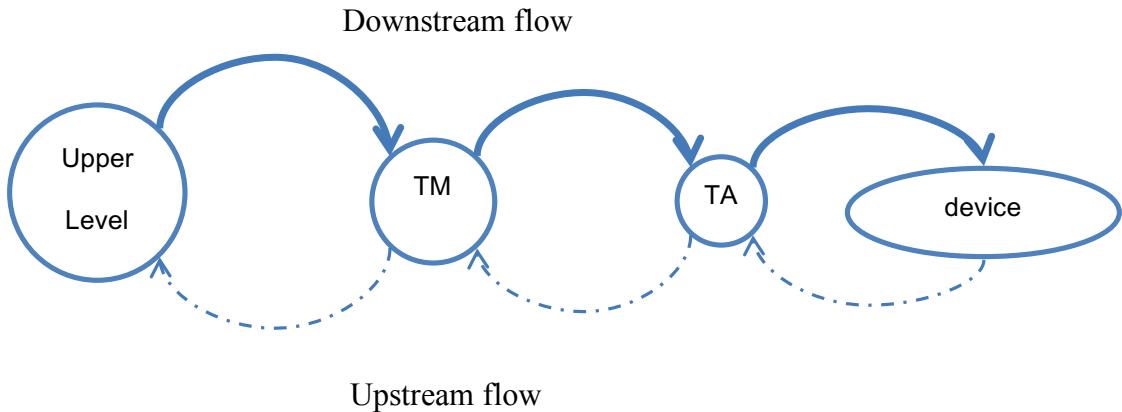


Figure 1: Transport Level Structure Diagram



*Figure 2: Transport level control and data flow
(downstream – to device, upstream – from device)*

The *Figure 2* demonstrates the data flows:

- Each abstraction level communicates only with the next and the previous abstraction level.
- Direct communication of two abstraction levels that are not the direct neighbors is prohibited.

For example:

The “Upper Level” cannot access the “Transport Adapter (TA)”

The “Transport Manager (TM)” cannot access the device and so forth.

The structure of the Transport Manager and the Transport Adapter is described in the below sections

3.1.2.1 Transport Manager Structure

The following diagram represents the structure of the Transport Manager.

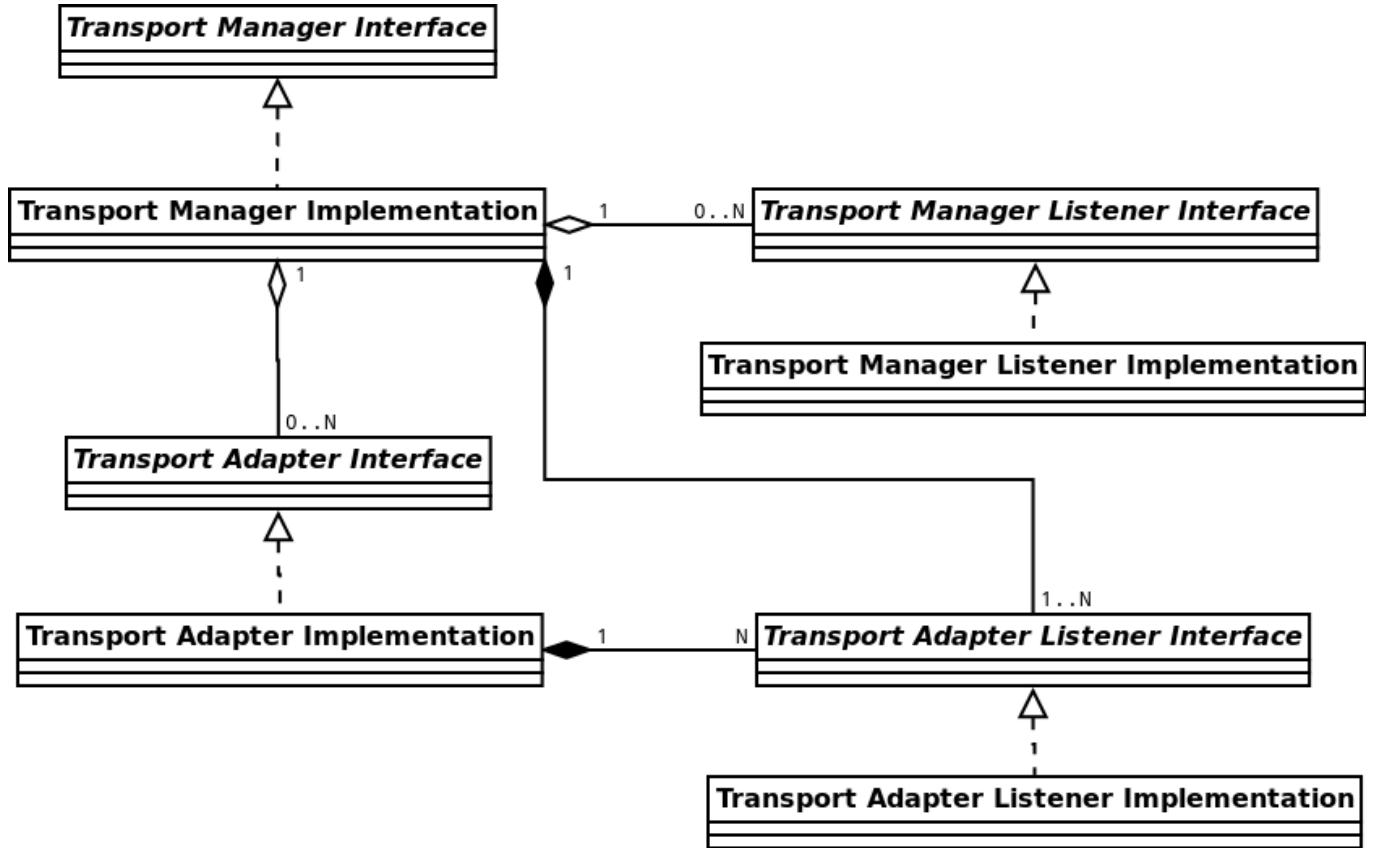


Figure 3: Transport Manager Class Structure Diagram

3.1.2.2 Transport Adapter Structure

The following diagram represents the structure of the Transport Adapter.

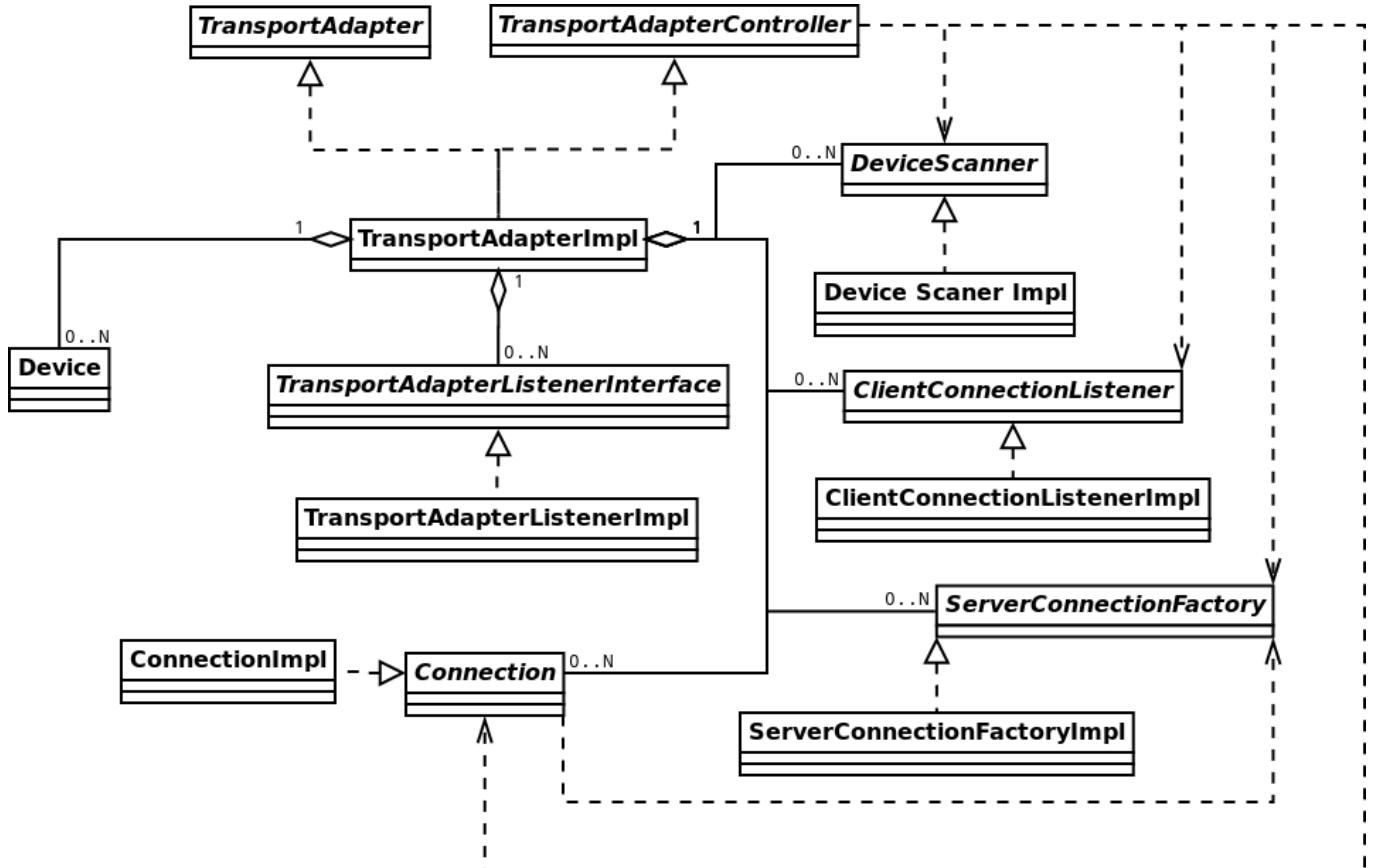
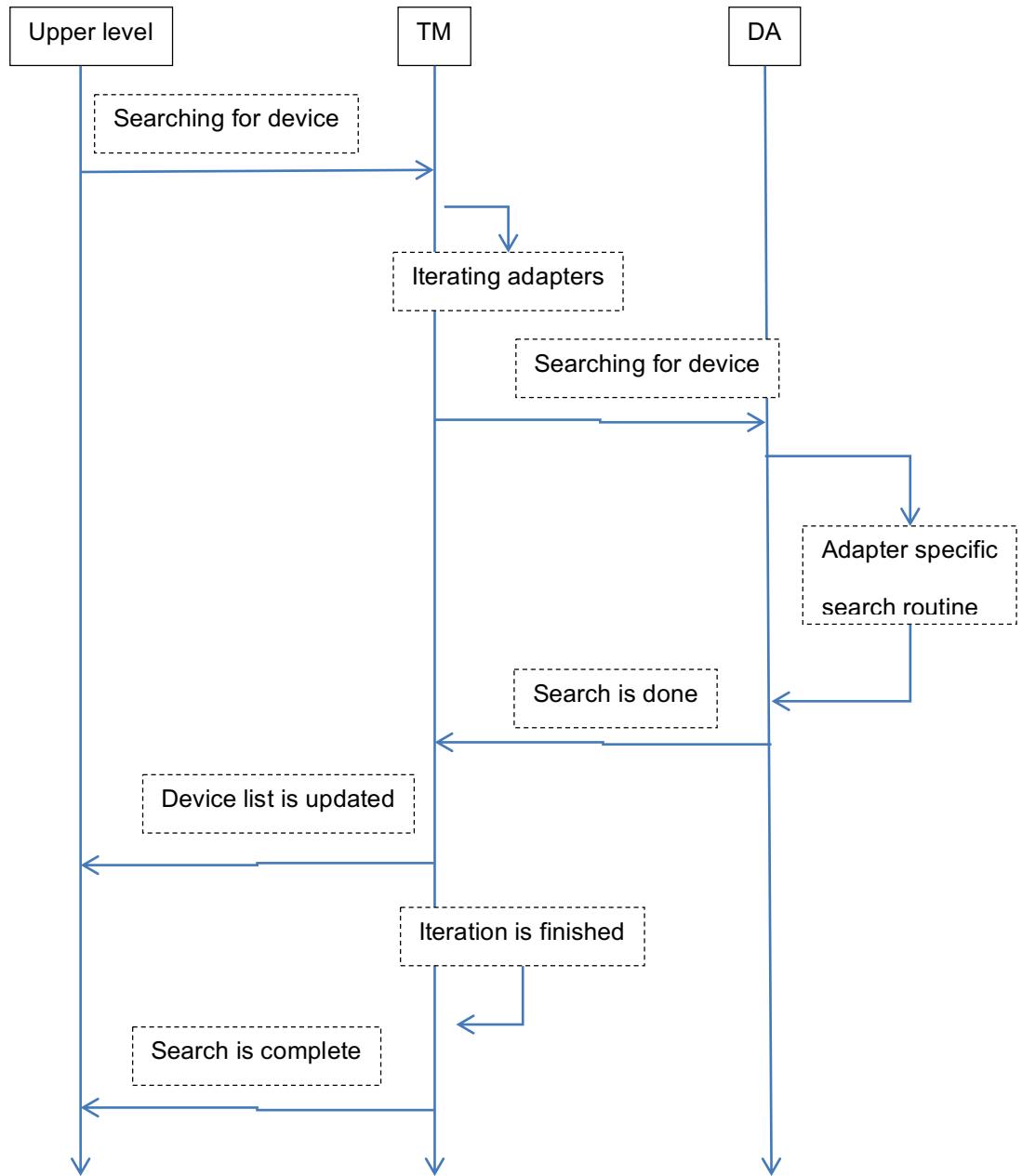


Figure 4: Transport Adapter Class Structure Diagram

3.1.3 Operation Examples (Message Sequence Chart)

3.1.3.1 New device search



*Figure 5: Transport Manager Message Sequence Chart
for a New Device Search*

3.1.3.2 Device-originating connection

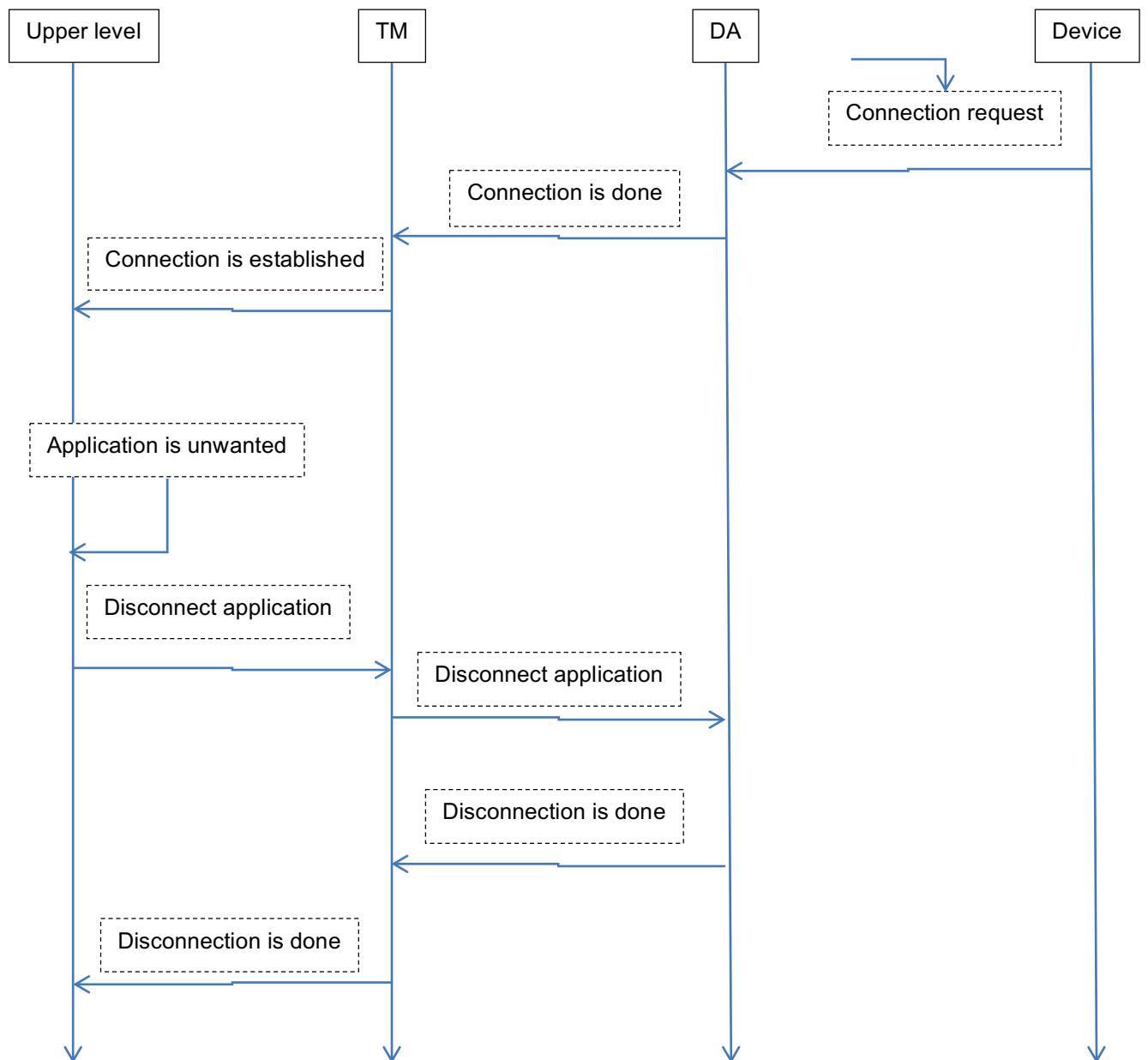


Figure 6: Transport Manager Message Sequence Chart for Device-Originated Connection

3.1.3.3 Connection close command

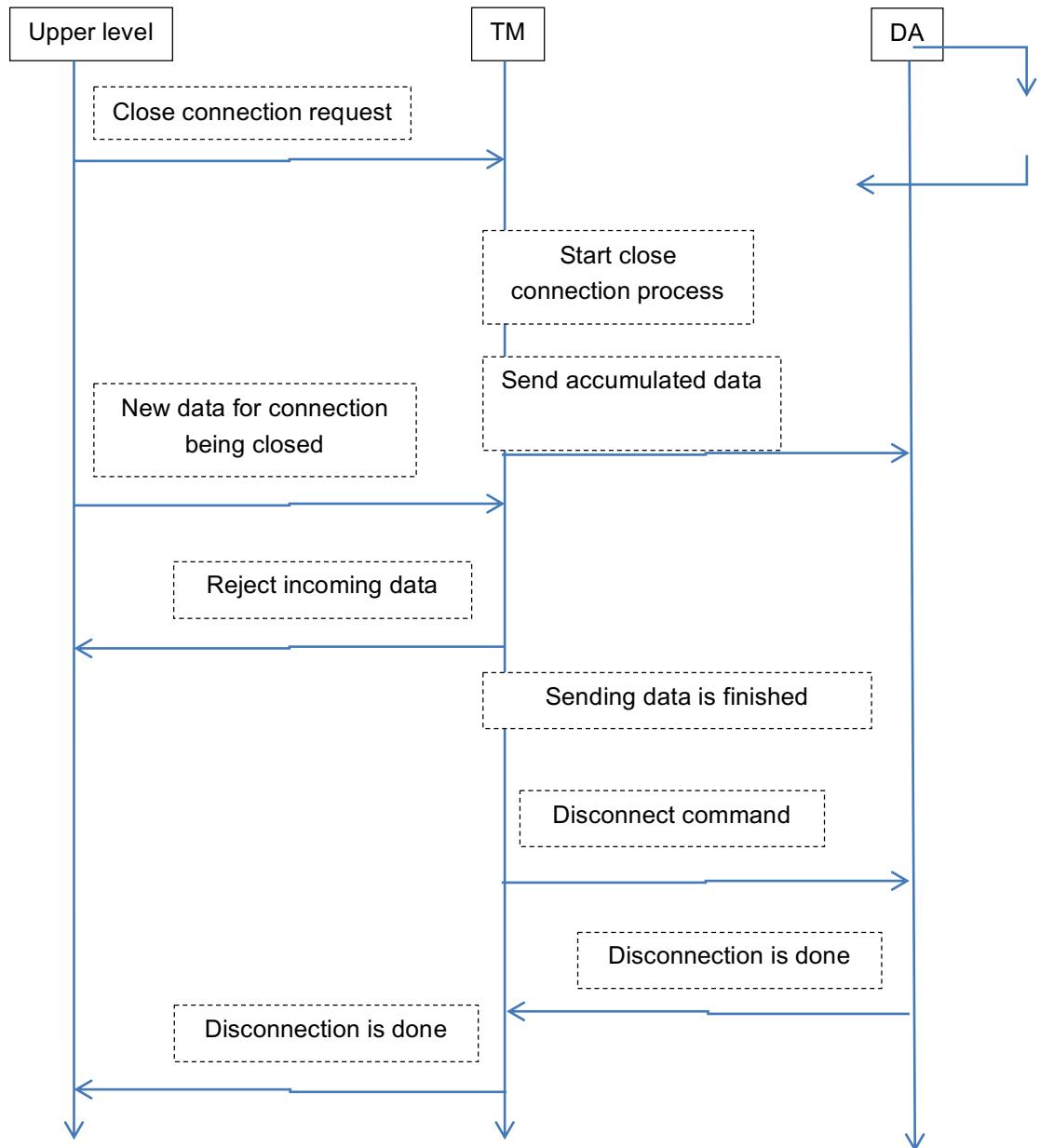


Figure 7: Transport Manager Message Sequence Chart for Connection Close Command

3.2 Transport Manager (TM) Usage

3.2.1 General Processing Description

3.2.1.1 *TM initialization*

- Every time SDL starts working it calls the creation and initialization of TM (the default or developer-defined one).
- TM uses singleton pattern and the instance of TM is created and initialized while being retrieved.
And the pointer is provided to the developer.
- If TM is initialized once it must not be initialized again.
- The developer may initialize the customized TM by calling the appropriate ‘init()’ function:
 - If the customized TM is based on default implementation, ‘init()’ must NOT be called twice for one and the same TM
 - If the customized TM is created from scratch, it is up to developer to choose the initialization mechanism.
- During the initialization process TM creates two threads:
 - For processing the message queue of commands from the upper level
 - For processing the events coming from devices.
- Also TM creates and initializes all available Transport Adapters (the default and/or customized ones).
- If appropriately configured TM may load the information about previous state and perform necessary actions (e.g. reconnect the last connected application).
- TM becomes initialized:
 - When its internal threads are created and are ready for working (i.e. the ‘init()’ function is serial and if it returns the control, the initialization has completed successfully and TM is ready for working).
 - And it does not watch whether the Transport Adapter(s) has initialized by this time:
 - If any of TAs failed to initialize, this is informed into the log file. TM answers with erroneous messages to the requests related to such TA.
 - If there are NO TAs (either not initialized or not defined), TM

answers with the error messages to all of the requests.

3.2.1.2 *TM structure*

- *TM Core*: it is a TransportManagerImpl class. It contains the processing mechanisms (e.g., for procedures of sending/receiving the data, for connecting/disconnecting procedures, etc.). It does not have the embedded default Adapters and Listeners. And it would not work apart of the Wrapper.
- *TM Wrapper*: it is implemented in TransportManagerDefault class. It adds the default Transport Adapter(s) and Listener to the Core completing the fully-featured functionality.

3.2.1.3 *TA initialization*

- The default Adapters are created and initialized by the default TM Wrapper.
- The customized Adapters should be initialized by the developer himself:
 - TM is initialized first
 - Then the init() function for the custom Adapter is called
 - Then the initialized custom TA and the Listener are added to the initialized TM.
- Transport adapter in its turn creates and initializes all available workers.
 - ‘Search Device’ worker creates a separate thread and waits for the ‘Start’ command.
 - ‘Client Listener’ creates a separate thread and waits for connections from devices.
 - ‘Server Connection’ worker executes all actions on caller’s thread (does not create a thread).

3.2.1.4 *Getting started*

- When the initialization is complete, TM starts waiting for:
 - The user’s command
 - The device connection.
 - Resumption from “last state” singleton (if such resumption is specified via profile)
- When one of the above happens, TA:
 - Creates a separate thread for each connection with device’ application(s)
 - Notifies TM on connection created.
- When the connection is established, the Upper Level:

- May start a handshake routine with the application and then notify the user about the application is connected.
- May close the connection by sending an appropriate command to TM if the application or device is unwanted.

3.2.1.5 Errors in TM

- ‘*Immediate*’ error:
 - When TM is not able to execute any command it will immediately return the appropriate error code. For example: when `connect (app_id)` is called and TM is not initialized yet this type of error occurs.
- ‘*Postponed*’ error:
 - TM is able to execute a command and there is some error occurred in downstream.
 - The component, where the error has occurred, sends the appropriate information to TM.
 - Then TM provides a notification to the Upper Level.

3.2.1.6 Messages in TM

- TM is ready to send and receive data after the connection is opened,
- Sending messages:
 - Each message destined for the device is posted into the message queue.
 - The message is removed from the queue after it is successfully sent to the device.
 - The message(s) is returned to the caller if for some reason TM is not able to send it (e.g., unexpected device disconnection),
- Receiving messages:
 - TM redirects messages from the Transport Adapter to the Upper Level via notification mechanism.

3.2.1.7 Connection identifiers

- TM uses the pair “Device ID” and “Application ID” for accessing the application on a device and internally for connection establishing.
- Device ID:
 - Stands for global unique identifier based on MAC address for network adapters and MAC address like for BT.
 - In the default implementation Device ID is logically split into two parts:

- Internal device ID – that is a MAC address string
 - External device ID – that is an integer value.
 - When the new MAC address is found, the integer value is assigned to it correspondingly (starting from 1 and incremented with every new assignment).
 - For persistent connection,
 - When the device is marked as “known”, the correspondence of internal and external IDs is stored (even after power off) until the user explicitly asks to “forget about <this> device”.
 - When the user requests to ‘forget’ the known device, the MAC address may be assigned with the new integer value on being connected for the next time.
 - For one-time connection this correspondence is not saved in long term storage. And if connected later the same MAC address may get the new integer value in correspondence.
 - It is not defined what happens if two devices with the same MAC address would connect.
- Application ID: is the application unique identifier. It is the incremental integer value assigned and used internally in TM.
- Connection ID:
- Connection is represented with a unique pair of “Device ID” and “Application ID”.
 - Connection ID is a system wide unique identifier (incremented integer value) assigned to each new connection.
 - This ID is used for sending/receiving data and for closing the connection.
 - There are no certain rules to define how exactly this ID is assigned.

3.2.1.8 *Connection closing*

- When TM receives ‘close connection’ or ‘disconnect’ commands it tries to finish sending accumulated messages and then closes the connection.
- If connection is lost TM will drain all accumulated messages and confirm connection closure.

3.2.1.9 *SDL shutting down*

- When SDL is going to shutdown, TM clears all objects which it has created (e.g. default transport adapters).
- Objects created by developer (e.g. developer's transport adapter or listener) are not removed by TM. The developer must take care of destructing his customized objects by himself.

3.2.2 Create Default TM Instance

For creating the instance of TM with default configuration it is necessary to use the following code:

```
#include "transport_manager.h"  
#include "transport_manager_default.h"  
  
{  
...  
    TransportManager* transport_manager =  
        TransportManagerDefault::instance();  
...  
}
```

Initialization of TM may take some time due to threads creation and initialization routine. After all is set and done Transport Manager is ready to be used. Till that time any command will be rejected with the error code “NOT INITIALIZED” in return value.

TM implementation uses singleton pattern, thus only one instance of TM will exist at a time. Nevertheless, this is not the rule and may be changed in custom TM.

3.2.3 Add Custom Listener to TM

- A Listener allows monitoring the events that take place in TM.
- The number of TM Listeners is not limited and may be zero.
- These listeners are provided to TM and are used by any module that needs to receive the notifications from TM.
- The list of Listeners that are called upon any event occurred in TM is stored in TM.
- TM does not create the own listener by default.
- Custom Listener:

- Its logic should be related with implementation of the module that uses this Listener.
- Should implement Transport Manager Listener Interface.

To set up the Listener the following code should be used:

```
#include "transport_manager.h"

#include "transport_manager_default.h"

class MyTransportManagerListenerImpl : public TransportManagerListener
{
    //implement here entire interface
}

{

...
TransportManager *tm_impl = TransportManagerDefault::instance();
TransportManagerListener *my_tm_listener = new MyTransportManagerListenerImpl();
tm_impl->addEventListenet(my_tm_listener);
...
}
```

To remove the Listener `removeEventListener` should be used.

3.2.4 Add Custom Transport Adapter to Default TM

3.2.4.1 About TA in general

- TM may contain zero to N Transport Adapters (TM with zero Adapters returns the erroneous messages for all of the requests).
- Each Adapter corresponds to specific type of transport (e.g. Bluetooth, LAN, USB, etc.).
- TA implements transport specific search, connect, disconnect and data transfer routines.

3.2.4.2 Several instances of TA

- It is allowed to have several adapters of the same type in one TM.
- The results of using several instances of default TAs are not defined.

- It is developer's responsibility to create a custom Adapter that operates well under conditions of using several instances of it.

3.2.4.3 Add Custom TA

- The simplest way to add a custom Transport Adapter is to derive the existing implementation of TCP or BT Adapter, adjust some behavior in derived class and add it to TM:

```
#include "transport_manager.h"

#include "transport_manager_default.h"

{

...

//note: the default implementation of TCP adapter is used in this example.

//developer is free to create his own implementation of transport adapter from scratch

TransportAdapter* my_transport_adapter =
    static_cast<TransportAdapter*>(new TcpTransportAdapter);

my_transport_adapter->init(); //note: TA must be initialized before getting instance of TM.

TransportManager *tm_impl =
    TransportManagerDefault::instance();

//note: when custom TA will be added to TM, the default listener will be assigned to it automatically

tm_impl->addTransportAdapter(my_transport_adapter);

...

}
```

3.2.4.4 Initialization

- By default TA is initialized during TM initialization.
- If TA needs to be added to the TM already initialized, this TA should be initialized first.

Note:

- *TM has a TCP adapter by default. Adding the new instance of the same Adapter may result the unexpected behavior (because default Adapters are not designed for working in such configuration).*
- *The developer must change the standard behavior of TCP adapter to eliminate the potential problems.*

3.2.5 Custom TA implementation from scratch

If default implementation of adapter is not suitable for particular case one can create new adapter from scratch:

- Create custom class that implements transport adapter interface. Note that custom TA shall use defined interface `TransportAdapterListener` for notifying Transport Manager.
- The instance of transport adapter listener will be set by Transport Manager automatically when custom transport adapter is added. If custom adapter does not store its listener then adapter will not be able to notify Transport Manager about events such as `OnDataReceiveDone` or `OnConnectDone`.

```
#include "transport_adapter.h"

class MyTransportAdapterImpl : public TransportAdapter
{
    // implement all interface functions here
    // use TransportAdapterListener interface to notify transport manager about event
    // happening
}
```

- Add the following code where the Transport Manager is initialized

```

#include "transport_manager.h"

#include "my_transport_adapter_impl.h"

{

...

TransportAdapter* my_transport_adapter =
    static_cast< MyTransportAdapterImpl*>(new MyTransportAdapterImpl);

//note: TA must be initialized before getting instance of TM

my_transport_adapter->init();

TransportManager *tm_impl = TransportManagerDefault::instance();

//note: when custom TA will be added to TM, the default listener will be
assigned to it automatically

tm_impl->addTransportAdapter(my_transport_adapter);

...

}

```

All internal logic implementation is up to developer.

Developer is responsible to implement:

- Communication with device
- Notification of state changes
- Error handling and so forth.

3.2.6 Add a New Listener to TA

- A Listener allows monitoring the events that take place in TA.
- The number of TA Listeners is not limited.
- The list of Listeners that are called upon any event occurred in TA is stored in TA.
- Custom Listener for TA:
 - The developer can add the Listener to TA through the customizing procedure only.
I.e., the developer needs to create his own TA on the base of the default one, and then to add the Listener to it.
 - Should implement Transport Manager Listener Interface.

Important Note:

- *Working with TA Listener by-passing the TM is dangerous and may lead to the asynchronous behavior of TM and the Adapter.*
- *The custom Listener should be added only together with the new custom Adapter and/or the new custom Transport Manager.*

To set up the Listener the following code should be used:

```

#include "transport_manager.h"
#include "transport_manager_default.h"

class MyTransportAdapterListener : public TransportAdapterListenerImpl
{
    //customize listener here if necessary
}

...
//note: the default implementation of TCP adapter and the default implementation
of adapter listener are used in this example.

//developer is free to create his own implementation of transport adapter from
scratch

TransportAdapter* my_transport_adapter =
    static_cast<TransportAdapter*>(new TcpTransportAdapter);
my_transport_adapter->init();

TransportManager *tm_impl =
    TransportManagerDefault::instance();

TransportAdapterListener* my_ta_listener =
    new MyTransportAdapterListener(tm_impl);
my_transport_adapter->addListener(my_listener)
tm_impl->addTransportAdapter(my_transport_adapter);

...
}

```

The developer should implement
“TransportAdapterListener” interface with the
custom logic before the Listener can be used.

3.2.7 Create TM with Custom TAs Only (with No Default Adapter)

If for some reason the default Adapters are not good enough they can be replaced with developer's defined Adapter(s). To do this the developer

- Must implement the Adapter's Interface using own logic
- Provide this new adapter to TM:

```

#include "transport_manager.h"

#include "transport_manager_impl.h"


class MyTransportManager : public TransportManagerImpl {

    virtual int init();

    virtual ~MyTransportManager();

    MyTransportAdapter *my_adapter_;

    explicit MyTransportManager(const TransportManagerAttr &config)
        : TransportManagerImpl(config),
        my_adapter_ (nullptr){

    }

public:

    static MyTransportManager* instance();

};

//note: the implementation of all methods above is not defined here just for
making the code look simpler

//Obvious MyTransportAdapter should be created and initialized somewhere.

{
    ...
    ...
    TransportManager *tm_impl = MyTransportManagerImpl::instance();
    ...
}

```

More detailed information on custom Transport Adapter creation is provided in section 3.2.9**Error! Reference source not found..**

3.2.8 Transport Manager Customizing

3.2.8.1 Basic information

- TM is responsible for all complex logic and decisions, while Transport Adapter is a primitive entity that operates only with transport specifics.
- Communication interface between TM and TA:
 - TM sends a command to TA.
 - If TA is unable to execute this command it returns the error code not processing the command.

- Otherwise, TA starts executing the command. Then TA notifies TM on executing completion by using the appropriate callback function.

3.2.8.2 *Queues as a fundamental of TM*

- Message Queue: for commands coming from the Upper Level.
- Event Queue: for events coming from devices.

3.2.8.3 *Rules for developer*

Customizing TM, the developer:

- Should implement transport manager interface.
- Should use the Transport Adapter Interface and the Transport Listener Interface for making his implementation work with default Adapters and Listeners.
- Has two options:
 - Creating the TM from scratch.
 - Deriving from the default implementation:
 - If not particularly changed, the default Adapters and Listeners will be used.

3.2.9 *Transport Adapter Customizing*

3.2.9.1 *Basic information*

- TA is highly adaptable to any specific of a real transport.
- TA consists of
 - So called *worker classes* that perform a single operation (e.g., device search)
 - *Controller* that
 - Accumulates event handling from all worker classes,
 - Controls the state of all internal data and
 - Notifies the Upper Level via callbacks
 - *Internal data structures* that contain the information about the device, the connection and other necessary information.

3.2.9.2 *Workers of TA*

- Device Scanner:
- Implements transport dependent search procedure initiated by the appropriate command from the Transport Manager.
- It is developer's responsibility to implement this worker.
- It may be absent for transport types that do not support searching.

- When the device is found (and in the current default implementation when all the devices are found) this worker:
 - creates a notification and directs this notification to the Controller
 - The Controller notifies TM using the TA Listener
 - TM receives the notification and sends a command to TA for connecting all available applications
 - TM and TA perform a chain of notifications
 - TM notifies the Upper Level using TM Listener with the information on each application connected: the connection ID, the application name, the device name.
- Client Connection Listener:
 - Implements the transport dependent connection that was originated by device.
 - It is developer's responsibility to implement this worker.
 - If transport does not support such ability this worker may be absent.
 - Working procedure:
 - This worker waits for the connection of a mobile device.
 - When the connection request from any of the mobile devices arrives, TA establishes a connection (creates the data path) with this device and the Connection Listener sends a notification through the Controller to the Upper Level with the device and application IDs.
- Server Connection Factory:
 - Implements transport dependent connection that was originated by the user.
 - It is developer's responsibility to implement this worker.
 - If transport does not support such ability this worker may be absent.
 - Creates a connection with the device and the application specified by the user:
 - Both the device and the application must be already known to TA by this moment.
 - TA may know about the device after 'search' routine or after 'restore previous state' routine.
 - If device is not known Transport Adapter returns the error immediately.
 - When the connection is created TA sends a notification through the Controller to the Upper Level.

3.2.9.3 Connection

- The main responsibility of Client and Server connection workers is to create a **Connection**.
- Connection is the entity that is responsible for data transmitting between the core and the device.
- Workers and Connection they use are closely related.
- Customizing:
 - Custom implementation of Connection must be used in custom worker(s) only.
 - It is not possible to use other types of data exchange in default workers.
 - It is possible to use default Connection implementation in custom worker.
- Default connection implementation is based on sockets isolated in separate thread (threaded socket connection).
- Each transport specific worker shall use transport dependent initialization of threaded socket connection.
- If the default implementation is not convenient to developer he can create his own Connection implementing any suitable way of data exchanging (e.g. shared memory connection).
- In this case custom workers shall be also created.

3.2.9.4 Descriptor

Descriptor is used for manipulating with devices and connections inside of the adapter.

3.2.9.5 Create custom Transport Adapter

The create custom TA using the Adapter Concept provided by SDK, the developer should do the following:

- Create a class that is derived from "TransportAdapterImpl" and add the implementation of necessary virtual methods. Let it be "getDeviceType".

```
#include "transport_adapter_impl.h"

class MyTransportAdapter : public TransportAdapterImpl
{
protected:
    virtual DeviceType getDeviceType() const;
}
```

- Create a connection class deriving from "ThreadedSocketConnection" and implement virtual methods.

```
#include "transport_adapter_impl.h"

class MySocketConnection : public ThreadedSocketConnection
{
    virtual ~MySocketConnection();
protected:
    virtual bool establish(ConnectError** error);
}
```

- Create a class for the device that will be used by Controller to manage devices and implement all virtual methods.

```
#include "transport_adapter_impl.h"

class MyDevice : public Device
{
    virtual ~Device();

    virtual bool isSameAs(const Device* other_device) const;

    virtual ApplicationList getApplicationList() const;
}
```

- Create the necessary worker classes by deriving from appropriate basic worker and fill them with necessary functionality.

```
#include "transport_adapter_impl.h"

class MyDeviceScanner : public DeviceScanner
{

class MyServerConnectionFactory : public ServerConnectionFactory
{

class MyClientListener : public ClientConnectionListener
{
```

These workers should use connection and device created in previous steps. For instance scanner should add a list of devices to controller. This list will be used later when ‘connect’ request will be received. To create a data path the connection class should be used.

When connection actually starts it will update Controller with the pointer to this connection.

- When everything is created it is time to combine all together.

```
#include "transport_adapter_impl.h"

MyTransportAdapter::MyTransportAdapter()
    : TransportAdapterImpl(
        new MyDeviceScanner(),
        new MyServerConnectionFactory(),
        new MyClientListener())
{
}
```

- Create an instance of the new adapter and provide it to the Transport Manager.

```
#include "transport_manager.h"
#include "transport_manager_impl.h"
#include "my_transport_adapter_impl.h"

{
...
TransportAdapter* my_transport_adapter =
    static_cast<TransportAdapter*>(new MyTransportAdapter);
TransportManager *tm_impl =
    TransportManagerDefault::instance();
tm_impl-> addTransportAdapter(my_transport_adapter);
...
}
```

This adapter will use the default connection implementation and default Adapter Listener but three worker classes will implement developer's logic. Also Transport Adapter will provide device type in developer's defined way.

`TransportAdapterImpl` virtual (but not pure virtual) methods `Store()` and `Restore()` can be reimplemented to provide resumption mechanism. Default implementations for both methods do nothing.

4 Build and Run SDL and Set Up the Environment

This instruction contains the information on how to build and run SDL depending on target OS and HMI emulator type.

4.1 General

The following configurations are available:

Target OS	HMI emulator type	Section
Linux	HTML5	4.5.2

To turn media features (libraries of PulseAudio and GStreamer) on/off please set `EXTENDED_MEDIA_MODE` to ON/OFF, for example:

```
cmake -DEXTEDDED_MEDIA_MODE=OFF
```

4.2 Known issues

1. HTML5 HMI does not display icons in case it is running on a remote host (the reason: SDL sends absolute paths on local computer).
2. HTML5 HMI can not play video & audio from pipe.

4.3 Preparation steps

Please see section "4.1 General" to determine which of the below steps must be used.

4.3.1 To prepare the Linux Host

Note:

In case Ubuntu/Xubuntu 12.04 or higher is already installed, git repository is cloned and is switched to <Main_Develop> branch, proceed from chapter 4.5 according to required configuration.

Linux workspace preparation:

1. Install Ubuntu/Xubuntu 12.04 or higher
2. Configure source repositories (check "Canonical Partners" repositories in Update Manager->Settings->SW source).

Note: Fore Ubuntu 12.04 "Update Manager->Settings->Other Software"

3. Perform "sudo apt-get install git" command in terminal.

Clone HMI repository https://github.com/LuxoftSDL/sdl_hmi

Clone git repository with ".

https://github.com/smartdevicelink/sdl_core (genivi) or
https://github.com/CustomSDL/sdl_panasonic (panasonic)

Note: The permissions to download the `sdl_panasonic` are required. Apply `skoveshnikov@luxoft.com` on this matter.

4. Switch to current release branch: "git checkout "<Release tag>"".

Note: in above cases the <Project_Root_Src> is the path to 'Project_Root_Src' git folder (e.g. "/home/user_name/Work/Project_Root_src").

Note: Restart computer if necessary.

Note:

Read the built-in setup environment help ("./setup_env.sh -help") which provides the description of predefined options of setup packages to be installed.

Executing ./setup_env.sh without a key will result installing the packages necessary for building SDL under Linux, with HTML5 HMI.

4.3.2 To set up Android Simulator for using instead of real device

Install and configure Android Simulator according the following instructions (valid on 2015/08/11):

1. Check support of hardware virtualization

Open terminal in VM and execute:

```
egrep '(vmx|svm)' /proc/cpuinfo
```

If an output is **empty**, your CPU doesn't support hardware virtualization, otherwise it does,

Note: Make sure that virtualization is enabled in BIOS (see "Advanced BIOS features" -> "Intel Virtualization Technology").

2. Install Kernel-based Virtual Machine (KVM)

```
sudo apt-get install kvm libvirt-bin  
bridge-utils
```

Add the user to the group:

```
sudo adduser $USER libvирtd
```

To take an effect the group membership log out and log in back on Xubuntu.

To check if KVM has been installed successfully, run:

```
virsh -c qemu:///system list
```

It will display message like:

Id	Name	State

```
root@server1:~#
```

Run:

```
kvm-ok
```

In case the output confirms, acceleration can be used:

```
INFO: /dev/kvm exists
```

```
KVM acceleration can be used
```

Otherwise,

```
INFO: /dev/kvm does not exist  
HINT: sudo modprobe kvm_intel  
INFO: For more detailed results, you should  
run this as root
```

```
HINT: sudo /usr/sbin/kvm-ok
```

The module 'kvm_intel' must be loaded:

Note: make sure that virtualization is enabled in BIOS,
see "Advanced BIOS features" -> "Intel Virtualization
Technology":

```
sudo modprobe kvm_intel
```

3. Setup Android Virtual Device (AVD)

Download Android
SDK <http://developer.android.com/sdk/> (See
*DOWNLOAD FOR OTHER PLATFORMS SDK Tools
Only*)

Note: Android SDK version may be different.

Note: Android API revision must be 17 or higher.

To install Android SDK execute:

```
tar -xf android-sdk_r22.2.1-linux.tgz
```

```
cd android-sdk-linux/tools/
```

```
./android update sdk --no-ui --filter  
android-18,sysimg-18,platform-tools
```

In case proxy is used run:

```
./android update sdk --no-ui --filter android-18,sysimg-18,platform-tools --proxy-host YC
```

Infomation:

- 1) YOUR_PROXY_HOST is proxy host without protocol, e.g. spb-proxy.spb.luxoft.com
- 2) YOUR_PROXY_PORT is proxy port, e.g. 8080
- 3) Flag **--no-https** if uses HTTP instead of HTTPS

If the warning obtained, execute the command below and choose necessary component versions to update via Android SDK Manager GUI:

Warning: The package filter removed all packages. There is nothing to install.

```
Update again without a package filter:
```

```
./android update sdk
```

To create a virtual device:

```
./android create avd --name A18 --target  
android-18 --abi x86 --sdcard 512M --skin  
WVGA
```

5 Audio/Video Streaming over SDL

5.1 PulseAudio and GStreamer libraries: how to include or exclude from project

5.1.1 Short overview

PulseAudio library enables streaming audio through BlueTooth A2DP over SDL.

GStreamer library enables audio capturing from the HU's microphone over SDL.

5.1.2 Include

To have SDL project with both libraries included, `EXTENDED_MEDIA_MODE` flag must be set to `ON` when building the project (see section '*4.5 Build SDL from source on Linux Host* '):

```
-DEXTENDED_MEDIA_MODE=ON
```

5.1.3 Exclude

To have SDL project with both libraries excluded, `EXTENDED_MEDIA_MODE` flag must be set to `OFF` when building the project (see section '*4.5 Build SDL from source on Linux Host* '):

```
-DEXTENDED_MEDIA_MODE=OFF
```

5.2 Use SyncProxyTester to check audio/video streaming over SDL

1. Follow the appropriate instructions from chapter '*4 Build and Run SDL and Set Up the Environment*' to build SDL project (mind to set `EXTENDED_MEDIA_MODE` flag to `ON`).
2. Make sure SDL console logging is switched off (e.g. by directing them to file:
`./smartDeviceLinkCore > SDL.log`), as it is a known fact that logging lowers the performance.
3. Start SyncProxyTester (SPT) of the latest version on mobile device:
 - Check 'Mobile Navi' box when starting SPT (as only navigation applications are allowed to stream audio/video over SDL)
 - Use USB or WiFi transports only (as BlueTooth is not the target transport for the feature).
4. Once SPT is registered, activate it on HMI (choose SPT in the HMI list of registered applications).

5. From SPT on mobile device perform the following:
 - For audio streaming:
 - Check the ‘Turn On Audio Service’ box (unchecked by default): for SPT to apply for starting audio service. Once SPT receives SDL’s acknowledgement on audio service started, it enables the ‘Start Streaming’ button.
 - Press ‘Start Streaming’ button.
 - For video streaming:
 - Check the ‘Turn On Video Service’ box (unchecked by default): for SPT to apply for starting video service. Once SPT receives SDL’s acknowledgement on video service started, it enables the ‘Start Streaming’ button.
 - Press ‘Start Streaming’ button.
6. The audio and/or video streaming is performed and processed by SDL. Depending on configuration in *smartDeviceLink.ini* file, SDL may forward the received stream to file/ through socket/ through pipe. Please see section 5.2 for details.

5.3 Configuring audio/video streaming parameters in *smartDeviceLink.ini* file

The below describes the possible configurations in *smartDeviceLink.ini* file for performing audio/video streaming from SDL to HMI. All the updates in this file might be done after the project is built.

1. *smartDeviceLink.ini* file is stored in the same folder where the project executable is.
2. The name of the file where audio/video stream is saved to (see p.1.6):

[MEDIA MANAGER]

...

```
VideoStreamFile = video_stream_file
//the value after “=” might be changed
AudioStreamFile = audio_stream_file
//the value after “=” might be changed
...
```

3. Choosing socket or pipe:

- The default values (set when the project is built):

[MEDIA MANAGER]

...

```

;VideoStreamConsumer = socket
    //video streaming through socket is commented
;AudioStreamConsumer = socket
    //audio streaming through socket is commented
;VideoStreamConsumer = file
    //video streaming to file is commented
;AudioStreamConsumer = file
    //audio streaming to file is commented
VideoStreamConsumer = pipe
    //video streaming is performed through pipe
AudioStreamConsumer = pipe
    //audio streaming is performed through pipe
...

```

- The updates may look like the following:

```

[MEDIA MANAGER]
...
VideoStreamConsumer = socket
    //video streaming is performed through socket
AudioStreamConsumer = socket
    //audio streaming is performed through socket
;VideoStreamConsumer = pipe
    //video streaming through pipe is commented
;AudioStreamConsumer = pipe
    //audio streaming through pipe is commented
...

```

4. Pipe name:

```

[MEDIA MANAGER]
...
;Temp solution: if you change NamedPipePath also
change path to pipe in
src/components/qt_hmi/qml_model_qtXX/views/SDLN
avi.qml
NamedVideoPipePath =
/tmp/video_stream_pipe      //the value after
“=” might be changed
NamedAudioPipePath =
/tmp/audio_stream_pipe      //the value after
“=” might be changed
...

```

5. Socket port:

```

[HMI]
...
VideoStreamingPort = 5050          //the value
after “=” might be changed
AudioStreamingPort = 5080          //the value
after “=” might be changed

```

II SDL-HMI Integration Guide

1 Overview

1.1 Operation System

SDL is installed to vehicle HU operation system, which might be:

- Linux Ubuntu 14.04
- QNX
- Any POSIX-compliant system.

1.2 Transports Supported

SDL supports the following transports for communication with HMI:

- 1) WebSocket (more details please find in section 3)
- 2) D-Bus (more details please find in section 4)

To communicate with SDL over one of the above transports, the Transport Adapter needs to be created on HMI side with the responsibilities to:

- Establish a connection with SDL
- Send/receive messages of corresponding to chosen transport format
- Translate SDL commands to format/calls understandable by HMI components.

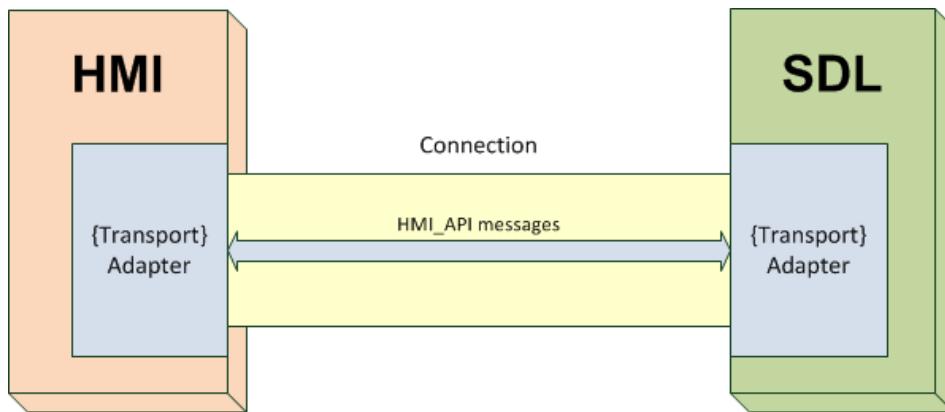


Figure 2.1 SDL-HMI Transport Adapters ({Transport} is the specific transport type).

The Transport Adapters of both WebSocket and D-Bus types are implemented in SDL.

For HMI it is enough to create one Adapter of the transport type chosen for communication.

Still, if there is a necessity, it is possible to expand the SDL with the Transport Adapter of another required type.

1.3 Communication

There are several requirements to HMI for performing the successful communication with SDL:

- 1) Establish a connection
- 2) Confirm its readiness to SDL
- 3) Correctly respond to the messages

2 WebSocket Transport

2.1 Connection Opening

2.1.1 Requirements to HMI adapter

Once decided to use WebSocket as a transport for communication between SDL and HMI, the below must be taken into consideration on HMI adapter creation.

HMI adapter must:

- Be installed on the same vehicle HU's OS where SDL is installed (Linux Ubuntu, QNX, or any POSIX-compliant system).
- Create and initialize the components named in correspondence to HMI_API (i.e. BasicCommunication, UI, Buttons, VR, TTS, Navigation, VehicleInfo).
- Establish a separate WebSocket connection with SDL for each of the above components (see 'Handshake' Example #1 and Example #3).
- To use the appropriate connection for sending requests/responses/notifications for the definite component (see Example #2).

2.1.2 Handshake

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

1. Client-Server Relations:
 - SDL is the Server (the one who confirms the WebSocket connection establishing)
 - HMI adapter is the Client (the one who requests the WebSocket connection opening).
2. Host:
 - When started and initialized SDL comes listening on:
127.0.0.1:8087
3. WebSocket Protocol:
 - [Version 13](#) is used.
4. Sequence of messages:
 - HMI adapter sends the handshake requests to SDL for every of the components (see Example #1 and Example #3). Each request must be sent:
 - To the mentioned host

- With the individual Sec-WebSocket-Key
- SDL provides the responses to every request with individual Sec-WebSocket-Accept.

2.1.3 Components registering

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

Here are the rules applicable to component registering messages only:

- "id" – a multiple of 100 integer value, e.g.:
 - 100 – for the first registering component
 - 200 – for the second one
 - 700 – for the seventh component
- "jsonrpc" : "2.0" - constant field and value for all JSON messages between SDL and HMI adapter.
- "method" :
 - "MB.registerComponent" – the request is assigned to SDL's MessageBroker (MB), 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 the response (Example 4.2) with:

- "id" – the value taken from the corresponding request.
- "result" – value of id multiplied by 10. Should be treated by HMI adapter as the success of registering.

2.1.4 Examples

Example #1 – WebSocket Handshake Headers

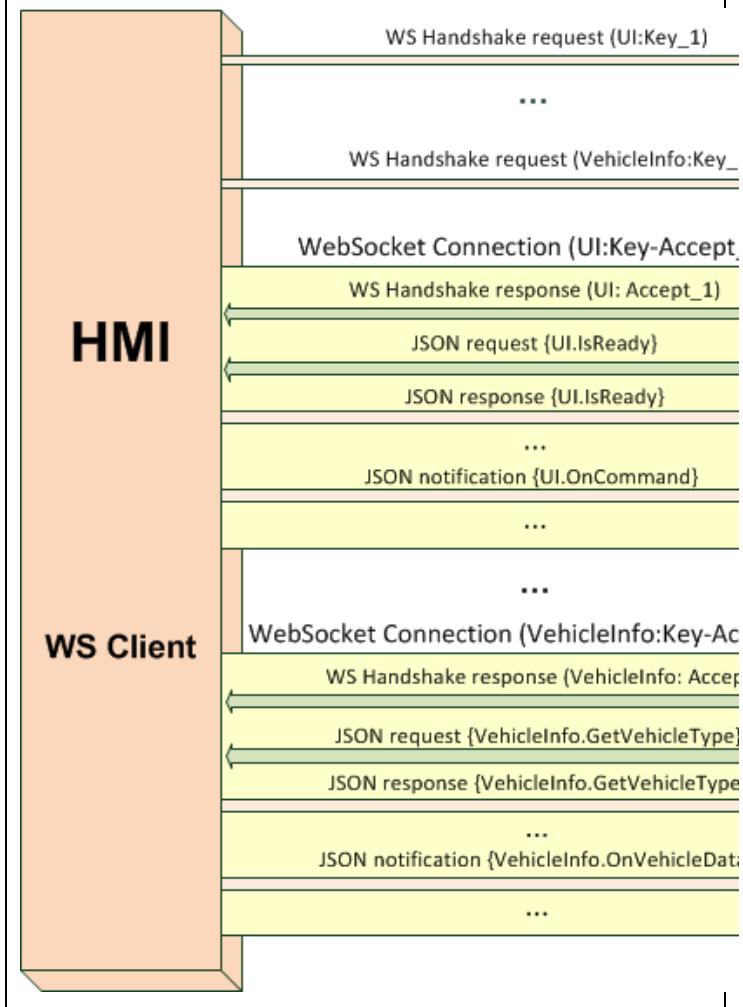
1.1 WS Handshake Request Header (sent from HMI)

```
GET / HTTP/1.1
Upgrade: websocket
Connection: Upgrade
Host: 127.0.0.1:8087
Origin: null
Pragma: no-cache
Cache-Control: no-cache
Sec-WebSocket-Key: b0atvDirYAt6OFfJ3DSGnw==
Sec-WebSocket-Version: 13
Sec-WebSocket-Extensions: x-webkit-deflate-frame
```

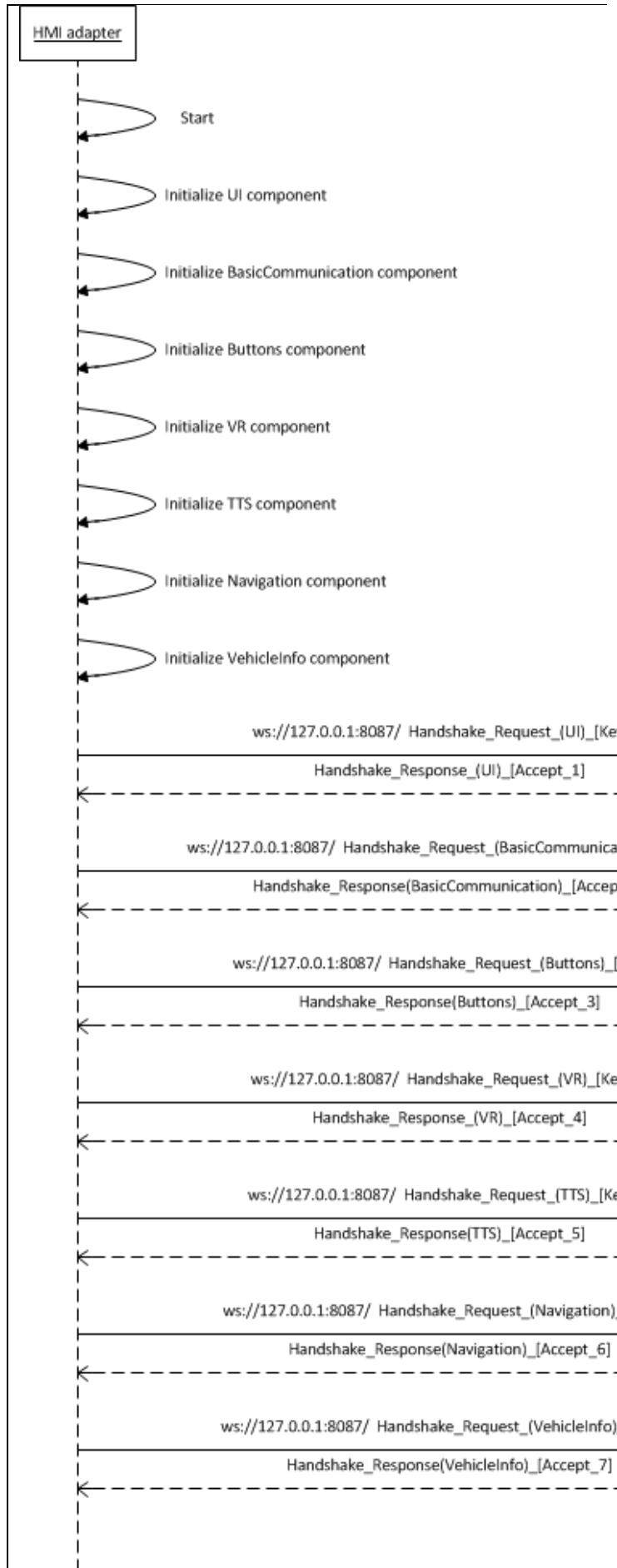
1.2 WS Handshake Response Header (sent from SDL)

```
Connection: Upgrade
Sec-WebSocket-Accept:
h195GcH5yhZwfS16giEqe9HiYRk=
Upgrade: WebSocket
```

Example #2 – WebSocket Connection Diagram



Example #3 – WebSocket Handshake sequence diagram



Example #4 – Register Component JSON Messages

4.1 Request

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

4.2 Response

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

2.2 JSON Message Format

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

Hereinafter within this chapter the actors for exchanging messages will be considered:

- The Client – the one who sends requests and notifications
- The Server – the one who provides the responses and sends notifications

Both SDL and HU components may act as a Client or as a Server.

2.2.1 Request

A RPC call is represented by sending a Request object to a Server. The Request object has the following members [1]:

Member name	Description
id	<p>An identifier established by the Client.</p> <p>This value MUST be of UNSIGNED INT type in the frames of communication between SDL and HU. This value should normally not be Null.</p> <p>If 'id' is not included the message is assumed to be a notification and should not be responded by the Server.</p>

Member name	Description
	(e.g. ""id" : 65382).
jsonrpc	<p>A String specifying the version of the JSON-RPC protocol. MUST be exactly "2.0":</p> <p>(e.g. "jsonrpc" : "2.0").</p>
method	<p>A String containing the information of the method to be invoked.</p> <p>The format is: "Component_Name.Method_Name"</p> <p>(e.g. "method" : "UI.IsReady").</p>
params	<p>A Structured value that holds the parameter values to be used during the invocation of the method. This member MAY be omitted.</p> <p>(e.g. "params" : { "cmdID" : 2318, "appID" : 409 }).</p>

The request might be sent with parameters or without any – please see the examples below.

2.2.1.1 Examples

Example #1 - A request with NO parameters:

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

Example #2 - A request WITH parameters:

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

2.2.2 Notification

A Notification is a Request object without an ‘id’ member. For all the other members that must be included to notification please see the [section 2.2](#).

The Server must not reply to a Notification, so no Response object needs to be returned to the Client.

2.2.2.1 Examples

Example #1 - A notification with NO parameters:

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

Example #2 - A notification WITH parameters:

Example #2 - A notification WITH parameters:

```
{  
    "jsonrpc" : "2.0",  
    "method" :  
"BasicCommunication.OnAppActivated",  
    "params" :  
    {  
        "appID" : 6578  
    }  
}  
  
{  
    "jsonrpc" : "2.0",  
    "method" : "Buttons.OnButtonPress",  
    "params" :  
    {  
        "mode" : "SHORT",  
        "name" : "OK"  
    }  
}
```

2.2.3 Response

On receipt of the Request message, the Server must reply with a Response (except for in the case of Notifications). The Response is expressed as a single JSON Object, with the following members

Member name	Description
id	This member is REQUIRED. It MUST be the same as the value of the 'id' member in the Request Object. If there was an error in detecting the id in the Request object (e.g. Parse error/Invalid Request), it MUST be Null. (e.g. ""id" : 65382).
jsonrpc	A String specifying the version of the JSON-RPC protocol. MUST be exactly "2.0": (e.g. "jsonrpc" : "2.0").

	<p>This member :</p> <ul style="list-style-type: none"> - is REQUIRED on success. - MUST NOT exist if there was an error invoking the method. <p>The 'result' field MUST contain:</p> <ul style="list-style-type: none"> - The 'method' field that has to include the same "Component_Name.Method_Name" as in originating Request. (e.g. "method" : "UI.IsReady"). - The 'code' field with '0' = SUCCESS value. No other result codes (described in the Result Enumeration in section 14.1.1) must be sent within the Response object. (e.g. "code" : 0). <p>This member MAY include the structured values determined by the method invoked on the Server. (e.g. "result" : <pre> { "available" : true, "code" : 0, "method" : "UI.IsReady" }).</pre> </p>
result	

2.2.3.1 Examples

Example #1 - A response with NO parameters (code and method are necessary):

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

Example #2: - A response WITH extra 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"
  }
}
```

2.2.4 Error

When a RPC call encounters an error, the Response Object MUST contain the ‘error’ member instead of the ‘result’ one. The error object has the following members:

Member name	Description
id	<p>This member is REQUIRED.</p> <p>It MUST be the same as the value of the ‘id’ member in the Request Object.</p> <p>If there was an error in detecting the id in the Request object (e.g. Parse error/Invalid Request), it MUST be Null.</p> <p>(e.g. ""id" : 65382).</p>

jsonrpc	A String specifying the version of the JSON-RPC protocol. MUST be exactly "2.0": (e.g. "jsonrpc" : "2.0").
error	<p>This member</p> <ul style="list-style-type: none"> - is REQUIRED on error. - MUST NOT exist if there was no error triggered during invocation. <p>The 'error' field MUST contain:</p> <ul style="list-style-type: none"> - The 'code' field with the value that indicates the error type that occurred. The result codes '1' – '23' from the Result Enumeration (described in section 5.1.1) should be used. (e.g. "code" : 14). - The 'message' field, containing the string that provides a short description of the error (from the same Result Enumeration). The message SHOULD be limited to a concise single sentence. (e.g. "message" : "There was a conflict with an already registered name"). - The 'data' field that MUST contain the 'method' member. The name in the form of "Component_Name.Method_Name" MUST be the same as in originating request. (e.g. "data" : { "method" : "UI.Alert" }).

2.2.4.1 Examples

Example #1 - Erroneous response
<pre>{ "id" : 103, "jsonrpc" : "2.0", "error" : { "code" : 13, "message" : "One of the provided IDs is not valid", "data" : { "method" : "VehicleInfo.GetDTCs" } } }</pre>

5 Getting Started

5.1 Readiness confirming

To start communication with SDL HMI must notify about its readiness via BasicCommunication.OnReady notification.

By receipt of OnReady notification SDL starts checking the availability of HMI components via the chain of RPCs:

- UI . IsReady –the display availability
- VR . IsReady –the voice recognition module availability
- TTS . IsReady – the Text-To-Speech module availability
- Navigation . Isready – check whether the navigation engine is available
- VehicleInfo . IsReady – check whether the vehicle information can be collected and provided.

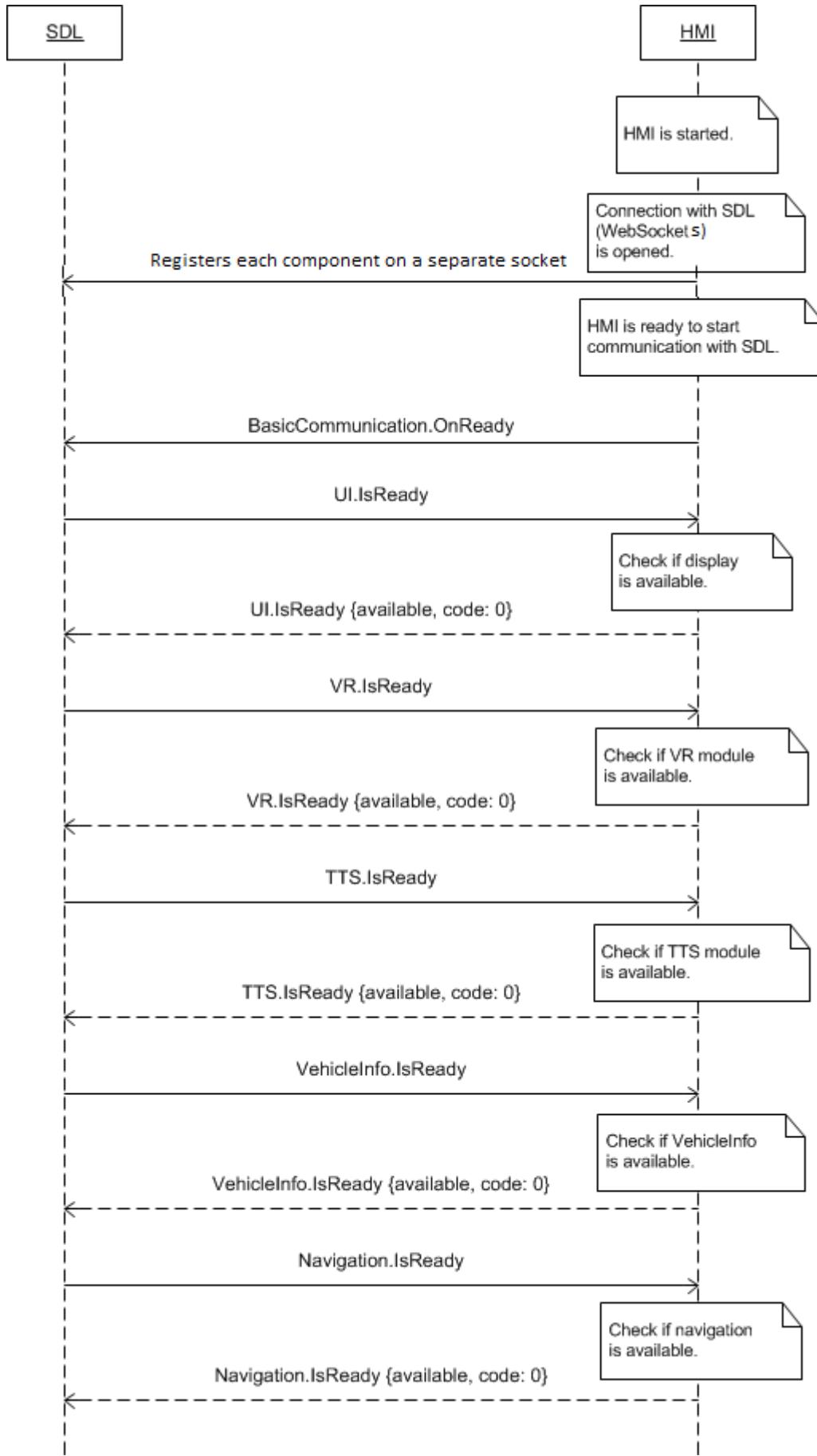
HMI must respond each of the above RPCs.

If the component is responded to be unavailable ("available": false), SDL will not further communicate with one.

5.1.1 Sequence diagram

Note:

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



6 BasicCommunication Component Description

6.1 UpdateDeviceList

6.1.1 Description

Type:	Function
Sender:	SDL
Purpose:	Update HMI's list of found devices.

The request comes after SDL has found a new device over one of available transports:

Notes about device searching:

With default SDL's Transport Manager (TM) and default Transport Adapters (TAs), SDL behaves in the following way:

Note:

Transport Manager as well as Transport Adapters may be customized or created from scratch, see chapter 3

1. BlueTooth transport:

- 1.1. SDL's TM makes a periodic search routine over BlueTooth.*
- 1.2. Once the device is found, SDL starts the procedure of searching SDL-enabled applications on such device.*
- 1.3. SDL sends BasicCommunication.UpdateDeviceList with the name and id of the discovered device to HMI.*
- 1.4. SDL needs to receive OnDeviceChosen(deviceInfo) or OnFindApplications(deviceInfo) from HMI to allow registering applications running on this device.*
- 1.5. After getting OnDeviceChosen(deviceInfo) or OnFindApplications(deviceInfo) from HMI SDL allows registering applications from the named device and sends BasicCommunication.OnAppRegistered to HMI.*

2. USB transport (AOA and iOS):

- 2.1. SDL learns about new device connected over USB (notification from TA over TM).*
- 2.2. SDL sends BasicCommunication.UpdateDeviceList with the name, id and TransportType (AOA/iOS) of the discovered device to HMI.*
 - a) In case iOS device was connected over USB HMI obtains UpdateDevice(TransportType=USB_IOS)*
 - b) In case Android device was connected over USB HMI obtains send UpdateDevice(TransportType=USB_AOA)*

2.3. SDL sends

BasicCommunication.OnAppRegistered to HMI right after (not waiting for OnDeviceChosen or OnFindApplications notifications).

For more details see diagrams [6.1.4.2 UpdateDeviceList \(TransportType=USB_IOS\)](#) and [6.1.4.3 UpdateDeviceList \(TransportType=USB_AOA\)](#)

3. WiFi transport:
 - 3.1. SDL learns about new device connected over WiFi (notification from TA over TM).
 - 2.2. SDL sends `BasicCommunication.UpdateDeviceList` with the name and id of the discovered device to HMI.
 - 2.3. SDL sends `BasicCommunication.OnAppRegistered` to HMI right after (not waiting for `OnDeviceChosen` or `OnFindApplications` notifications).

6.1.2 Request

6.1.2.1 Behavior

HMI must:

1. Update its list of found devices (that is, store the provided information).
2. Use this information later when providing the `OnDeviceChosen` or `OnFindApplications` notifications.
3. Provide the User with the possibility of choosing among the found devices through either display or voice recognition or both.

6.1.2.2 Parameters

Param Name	Type	Mandatory	Additional	Description
deviceList	<code>Common.DeviceInfo</code>	true	Array = true minsize = 0 maxsize = 100	The list of devices (name and ID) found. If contains the empty array, it means that all the devices have been disconnected or are not connected yet. See <code>DeviceInfo</code> .

6.1.2.3 DeviceInfo Structure

Param Name	Type	Mandatory	Additional	Description
name	String	true		The name of the device connected
id	String	true		The ID of the device connected. either hash of device's USB serial number (in case of USB connection) or hash of device's MAC address (in case of BlueTooth or WiFi connection). It remains unique between the ignition cycles for the same transport type .
transportType	Common.TransportType	false		The transport type the named-app's-device is connected over to HU (BlueTooth, USB or WiFi). Always returned by SDL in <code>OnAppRegistered</code> and <code>UpdateAppList</code> RPCs.

Param Name	Type	Mandatory	Additional	Description
isSDLAllowed	Boolean	false		Sent by SDL in UpdateDeviceList. 'true' – if device is allowed for PolicyTable Exchange; 'false' – if device is NOT allowed for PolicyTable Exchange

6.1.2.4 TransportType Enumeration

Element name	Value	Short Description
BLUETOOTH	0	
USB_IOS	1	
USB_AOA	2	
WIFI	3	

6.1.3 Response

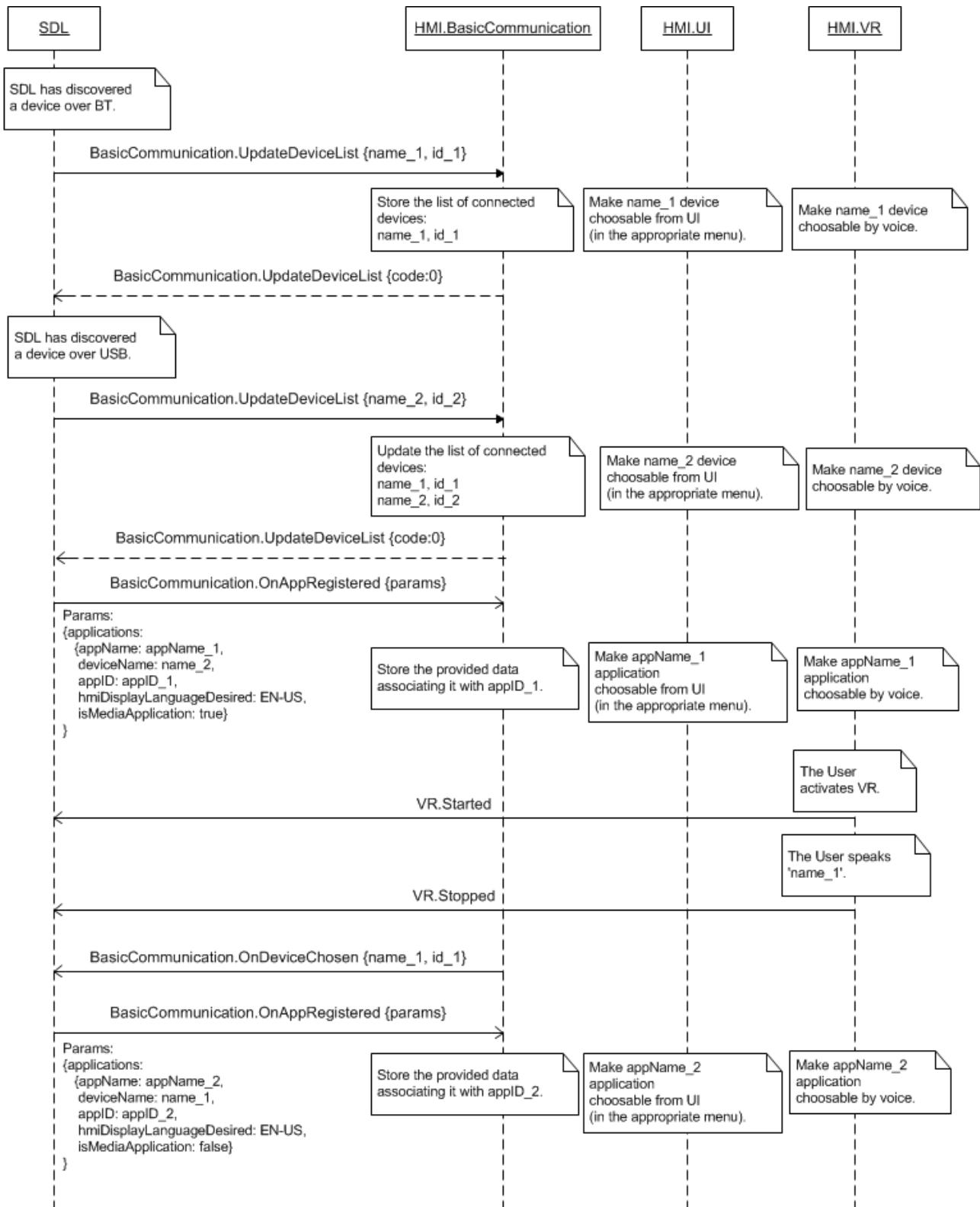
Note:

There is a difference in message type for WebSocket and D-Bus connection (see the table below).

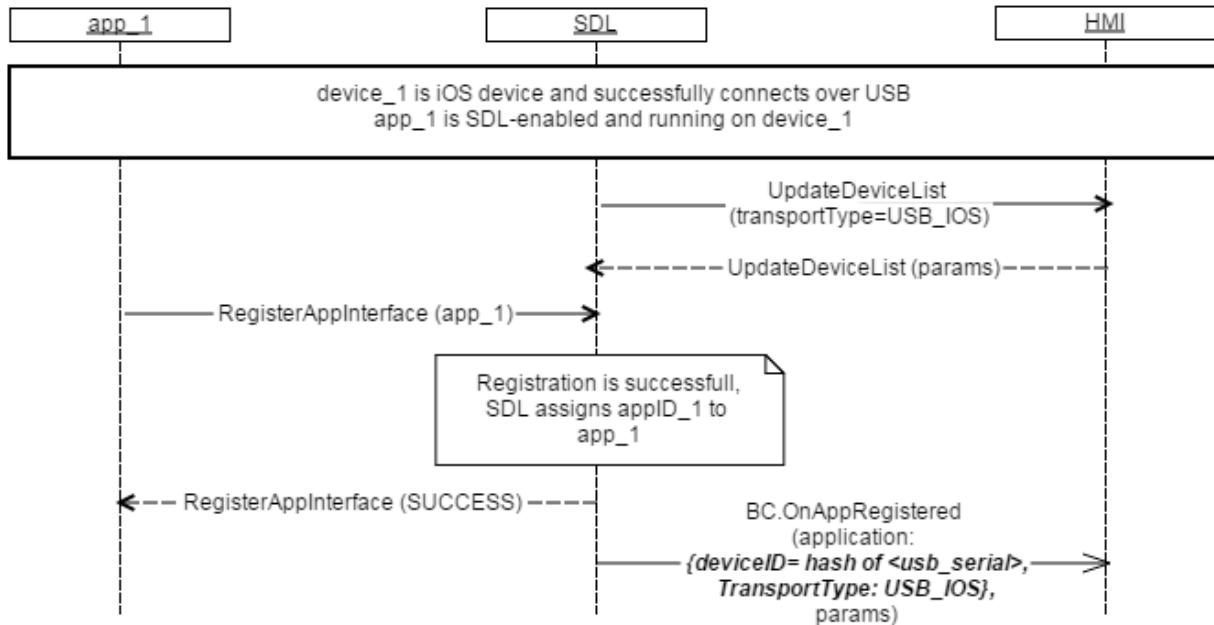
Result	Description	Message type		Message Params	Notes
		WebSocket	D-Bus		
Success	SUCCESS: HMI has stored the list of found devices provided within request.	JSON response	Regular response	code : 0	-
Failure	INVALID_DATA: The data sent is invalid (json, out of bound parameters etc)	JSON error message	Regular response	code : 11	Applicable for this RPC result codes. Please see Result Enumeration for all SDL-supported codes
	GENERIC_ERROR: The unknown issue occurred or other codes are not applicable.			code : 22	

6.1.4 Sequence Diagrams

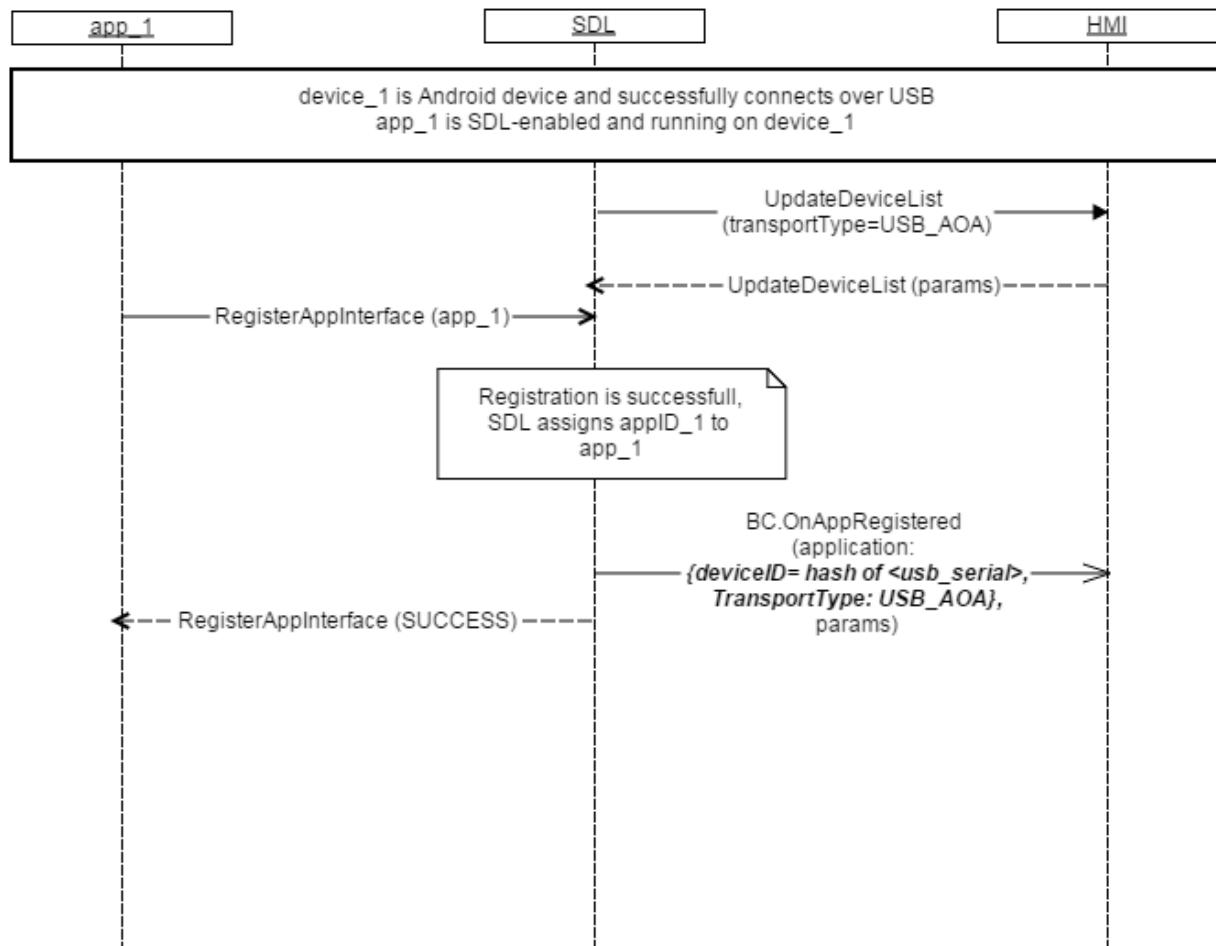
6.1.4.1 UpdateDeviceList after SDL finds a new device over BT and USB



6.1.4.2 UpdateDeviceList
(*TransportType=USB_IOS*)



6.1.4.3 UpdateDeviceList (TransportType=USB_AOA)



6.1.5 JSON Messages Examples

6.1.5.1 Request

```
{
    "id" : 64,
    "jsonrpc" : "2.0",
    "method" :
"BasicCommunication.UpdateDeviceList",
    "params" :
{
    "devicelist" :
[
    {
        "name" : "Jerry's Phone",
        "id" : 3
    },
    {
        "name" : "XT910",
        "id" : 4
    }
]
}
```

```
}
```

6.1.5.2 Response

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

6.1.5.3 Error message

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

6.2 ActivateApp

6.2.1 Description

Type:	Function
Sender:	SDL
Purpose:	Activate the named application on HMI.

Activated application is assumed to receive access to head unit's display, audio and control systems/modules:

- supported by the head unit
- confirmed to be supported via responses to `IsReady` requests sent by SDL to every of components (i.e. UI, VR, TTS, Buttons, VehicleInfo, Navigation).

The request may follow:

- After SDL restores application's previous-to-ignition-off state on HMI.

In case SDL resumes an app to FULL, SDL sends `BC.ActivateApp` to HMI(see [diagram 6.26.2.1 Resume AudioSource for the application in FULL](#) for more details).

In case SDL resumes an app to LIMITED, SDL sends `BC.OnResumeAudioSource` to HMI. See diagram [6.26.2.2 OnResumeAudioSource LIMITED HMILevel](#) for more details.

In case of successful HMI level resumption, SDL first assigns 'default_hmi' level from policies and after **3sec timeout**, resumes the application either to FULL or LIMITED (whichever applicable according to previous ignition cycle).

Note:

By the time ActivateApp comes, HMI is already provided with the detailed information on every registered application with OnAppRegistered RPC.

6.2.2 Request

6.2.2.1 Behavior

HMI must:

1. Activate the named application on HMI:
 - 1.1. Display the application-related screen.
 - 1.1.a. Display UI.Show related parameters associated with the named appID in case previously requested within ignition cycle..
 - 1.1.b. Display the corresponding template in case previously requested by UI.SetDispalyLayout for the named application (appID) since the application registration.
 - 1.1.c. Apply UI.SetGlobalProperties associated with the named appID in case previously requested since the application registration.
 - 1.1.d. Apply UI.AddCommand associated with the named appID in case previously requested since the application registration .
 - 1.1.e. Apply UI.AddSubMenu associated with the named appID in case previously requested since the application registration .
 - 3.2. Make VR commands accessible in case previously requested by VR.AddCommand for the named appID since the application registration.
 - 1.3. Apply TTS.SetGlobalProperties associated with the named appID in case previously requested since the application registration.
2. Assign priority based on the priority parameter received. If request comes with omitted parameter, HMI must assign the priority of NONE by default.
3. Respond with SUCCESS result code right after HMI activated the named application. For all of applicable to this RPC result codes please see section 6.2.3. Response.
4. To set-up the application as active audio source in case it is media/navigation type

6.2.2.2 Parameters

Param Name	Type	Mandatory	Description
appID	Integer	true	The ID of the application requested to be activated.
priority	Common.AppPriority	false	Send to HMI so that it can coordinate order of requests/notifications correspondingly.
level	Common.HMILevel	false	This parameter is omitted in case the app needs to be activated in FULL. The parameter is sent by SDL in case the app needs to be placed in one of HMI Levels LIMITED, BACKGROUND or NONE on HMI.

6.2.2.3 AppPriority

Element name	Value	Short Description
EMERGENCY	0	
NAVIGATION	1	
VOICE_COMMUNICATION	2	
COMMUNICATION	3	
NORMAL	4	
NONE	5	

6.2.2.4 HMILevel

Element name	Value	Short Description
FULL	0	An application have an access to HMI supported resources: UI, VR, TTS, audio system etc.
LIMITED	1	Application must be activated to LIMITED level, that is the application must be granted an access to audio system only.
BACKGROUND	2	Application must be activated inBACKGROUND level, that is the app must be listed in applications list, but have no acces to HMI supported resources Allowed to runRPCs according to Policy Table rules (managed by SDL).
NONE	3	App must be activated in NONE level, that is the app must be listed in applications list, but have no acces to HMI supported resources the same as not to run almost all RPCs (managed by Policy Manager).

6.2.3 Response

Note:

There is a difference in message type for WebSocket and D-Bus connection (see the table below).

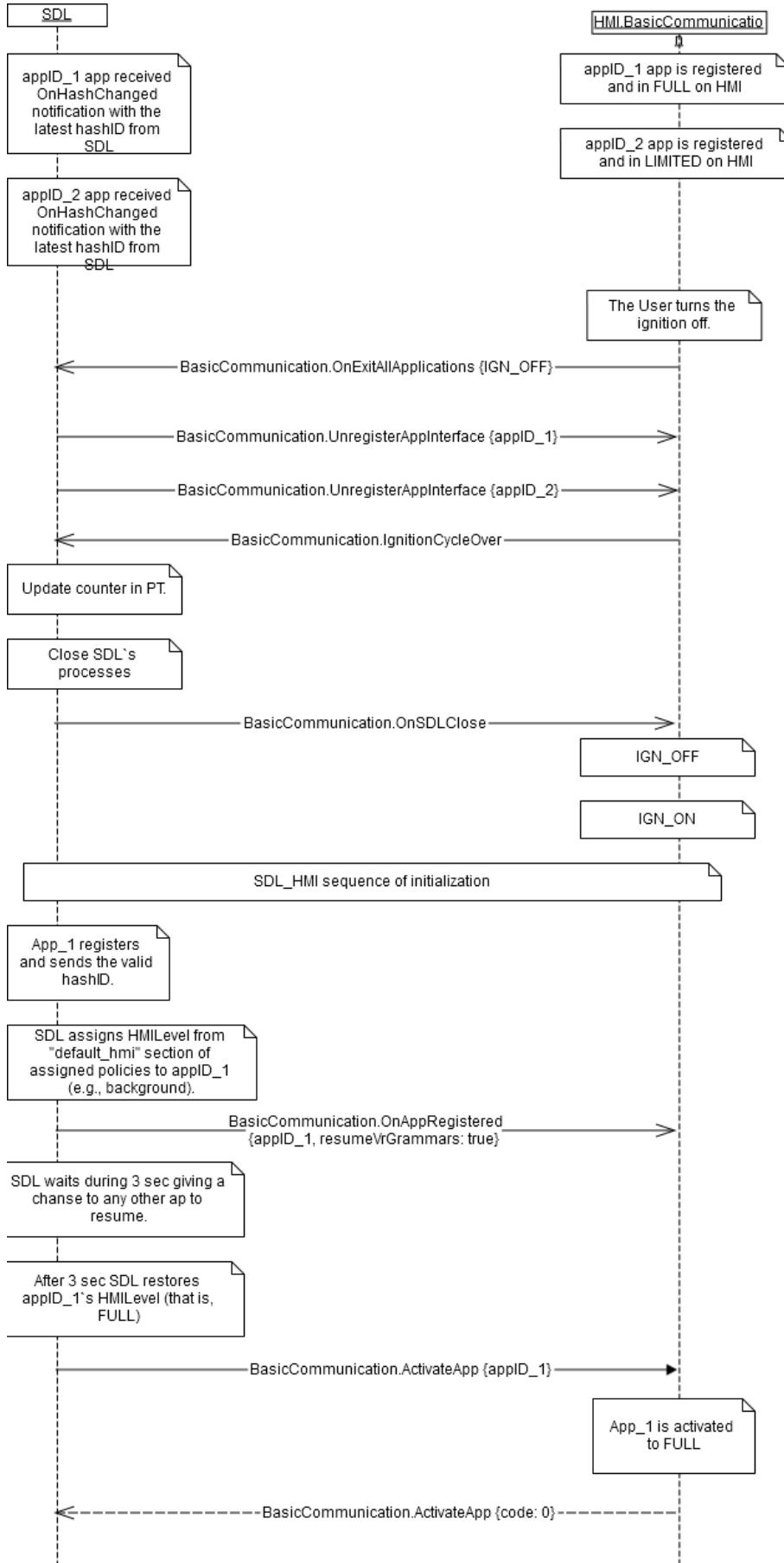
Result	Description	Message type		Message Params	Notes
		WebSocket	D-Bus		
Success	SUCCESS: HMI has activated the requested application (step 1. Of sub-section 6.2.2.1. is performed successfully)..	JSON response	Regular response	code : 0	-
Failure	INVALID_DATA Invalid json or parameters are out of bounds	JSON error message	Regular response	code : 11	Applicable for this RPC result codes. Please see Result Enumeration for all SDL-supported codes
	INVALID_ID Wrong appId (e.g. doesn't exist) HMI does not have the named application ID in its list of registered applications (that is, there were no OnAppRegistered with such appId from SDL previously).			code : 13	
	IGNORED: The named application is already in the same active mode on HMI			code : 6	
	GENERIC_ERROR: 1) The			code : 22	

	unknown issue occurred or other codes are not applicable.				
--	---	--	--	--	--

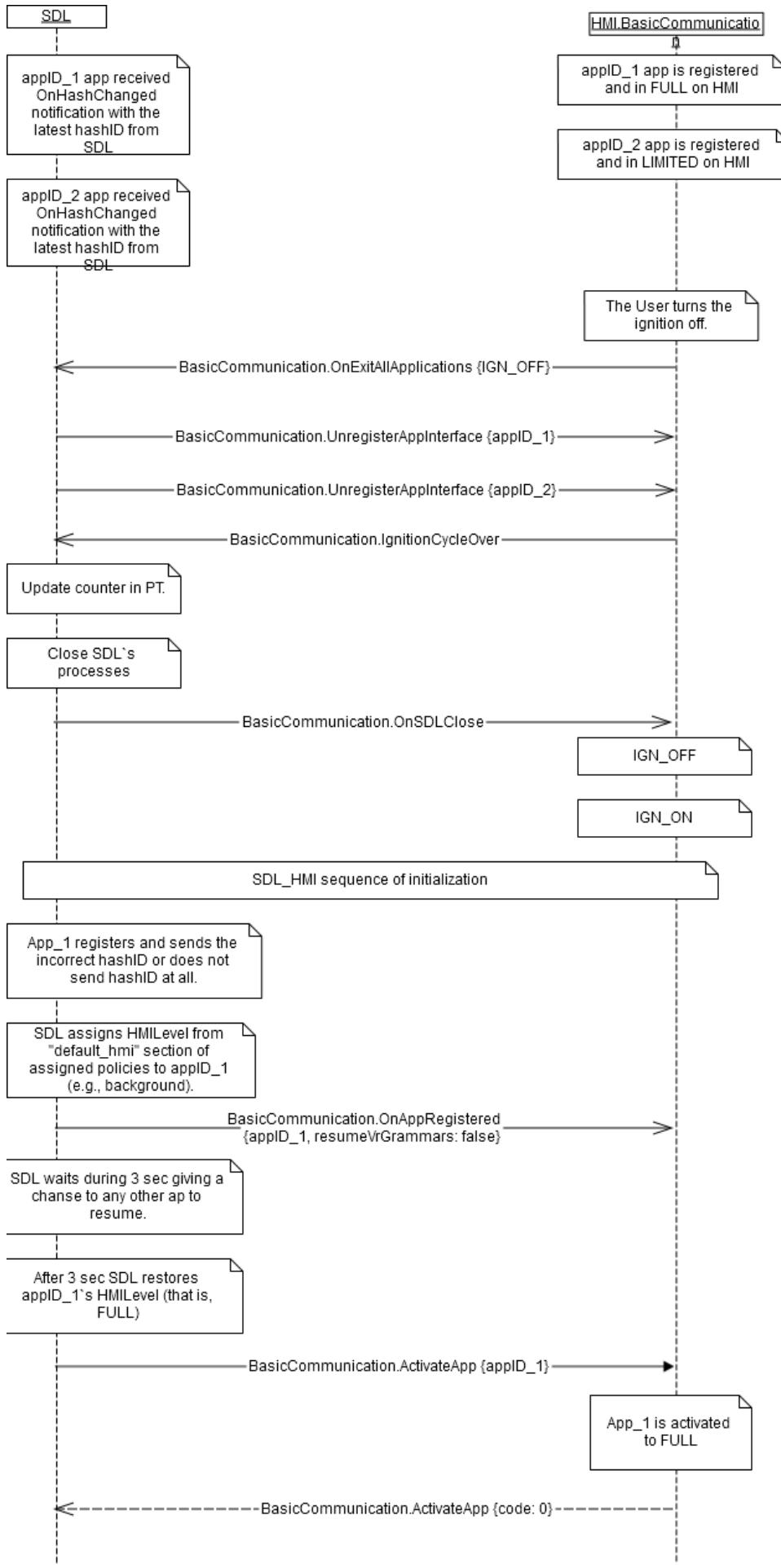
Note: In case HMI does not respond SDL's request during SDL-default timeout (10 sec), SDL will return `GENERIC_ERROR` result code to the corresponding mobile app's request.

6.2.4 Sequence Diagrams

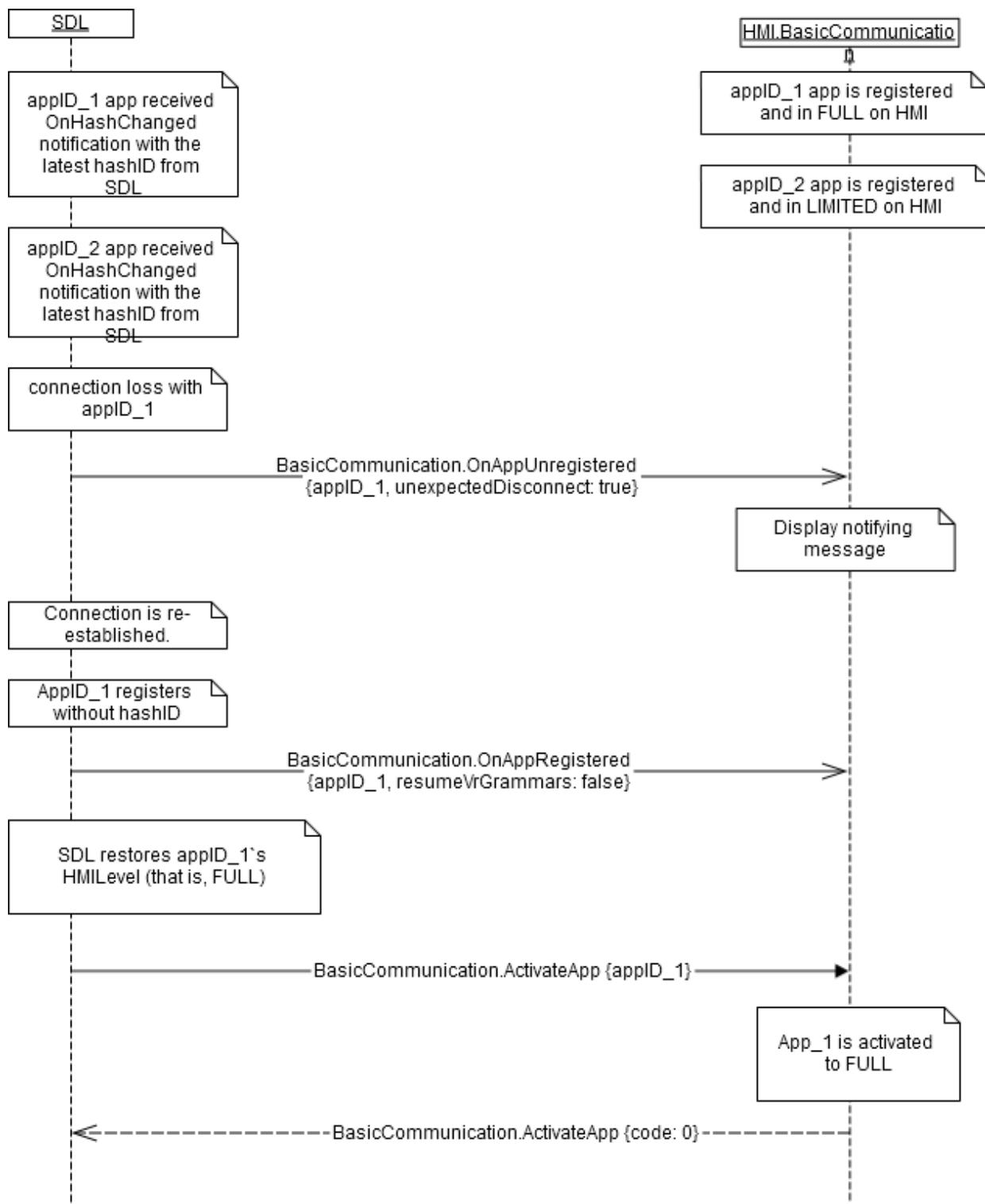
**6.2.4.1 BasicCommunication.ActivateApp after
successful resumption.**



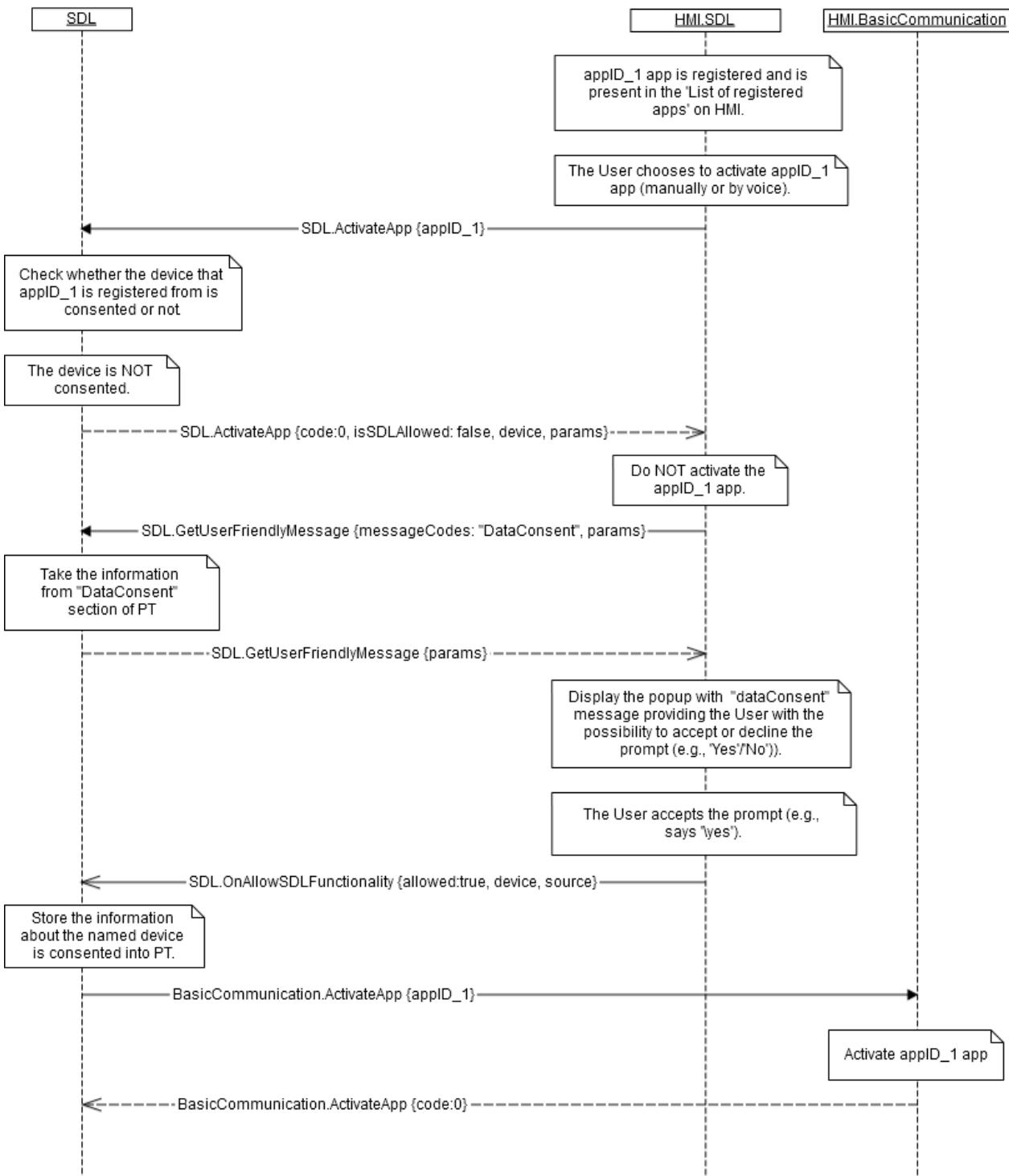
**6.2.4.2 BasicCommunication.ActivateApp after
UNsuccesssful resumption.**



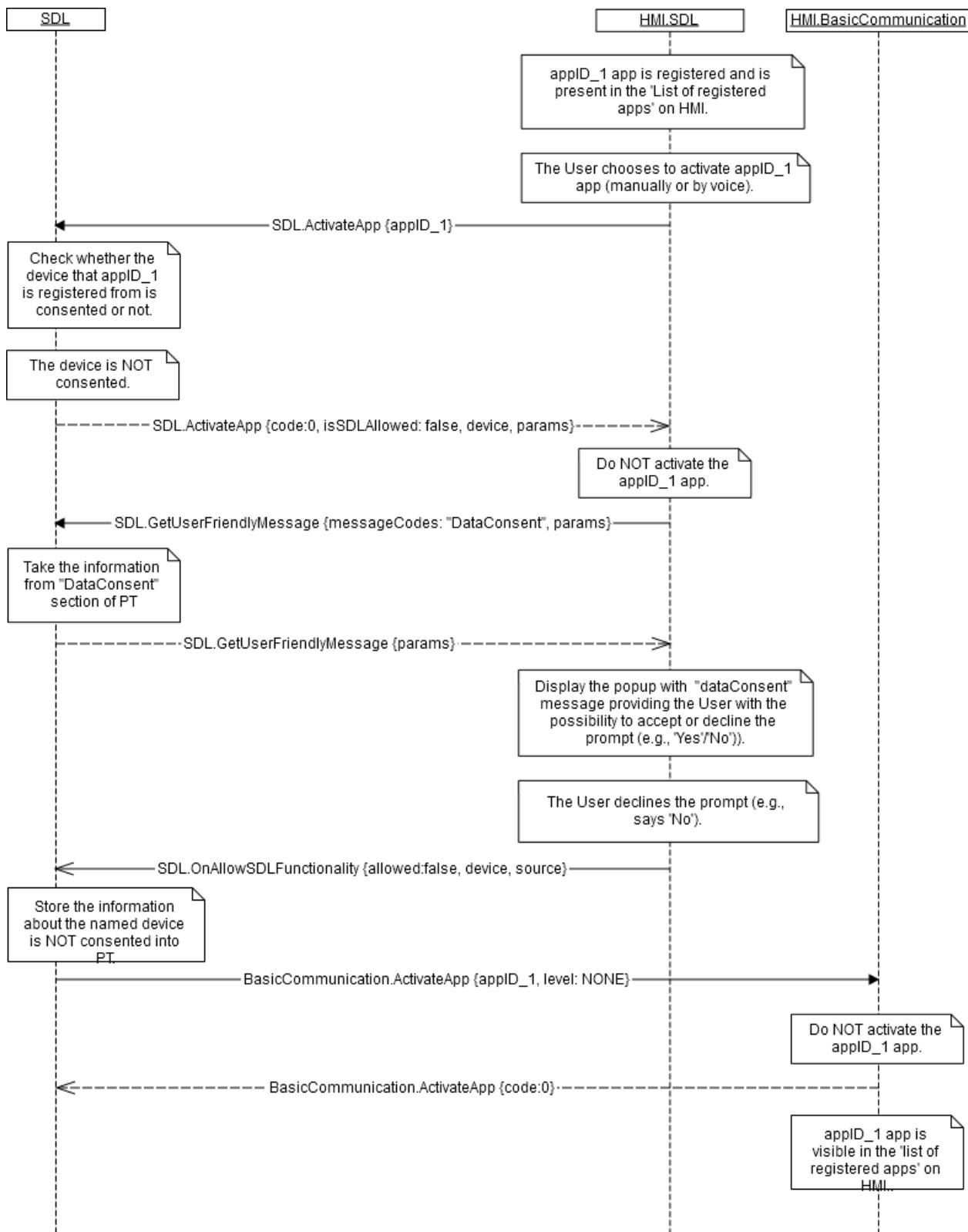
6.2.4.3 BasicCommunication.ActivateApp after unexpected disconnect



6.2.4.4 BasicCommunication.ActivateApp after the User accepted the data consent prompt



6.2.4.5 BasicCommunication.ActivateApp after the User declined the data consent prompt



6.2.5 JSON Messages Examples

6.2.5.1 Request

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

6.2.5.2 Response

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

6.2.5.3 Error message

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

6.3 MixingAudioSupported

6.3.1 Description

Type:	Function
Sender:	SDL
Purpose:	Find out if HMI supports mixing audio.

'Mixing audio' names the ability of vehicle audio system to speak the TTS prompts or to listen to and recognize the VR commands while playing audio.

Note:

SDL is able to get the information about ‘mixing audio’ capability of HMI from `smartDeviceLink.ini` file (this file is usually stored in the same folder where `SDL` executable is).

6.3.2 Request

6.3.2.1 Behavior

HMI must:

- 1) Check its mixing audio capabilities and provide the correct response.

Note:

If the request is NOT responded and there is no record in `smartDeviceLink.ini` file, `SDL` will consider that mixing audio is NOT supported.

6.3.3 Response

6.3.3.1 Behavior

Note:

There is a difference in message type for WebSocket and D-Bus connection (see the table below).

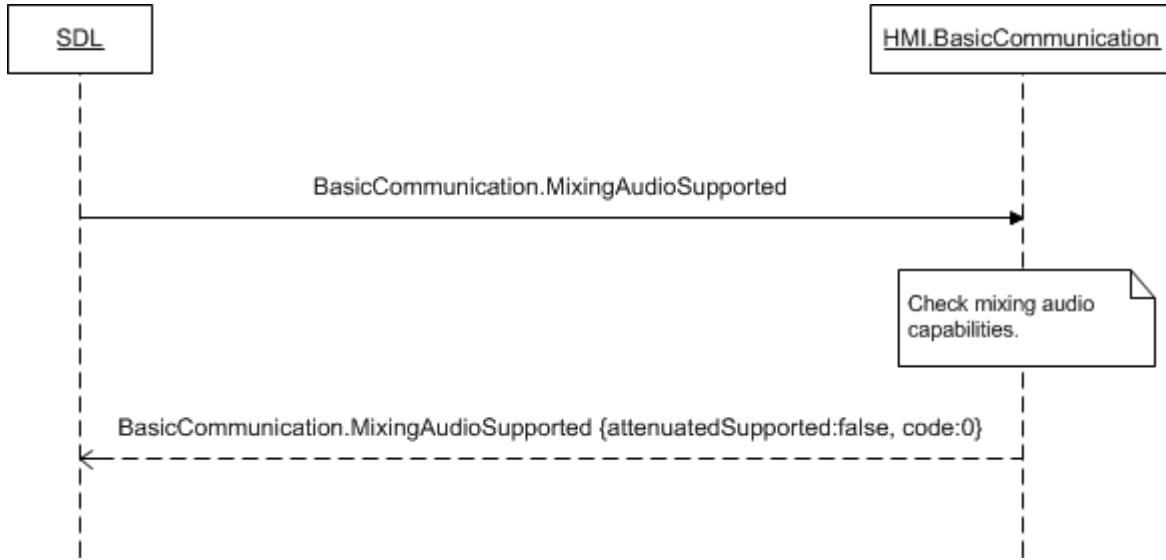
Result	Description	Message type		Message Params	Notes
		WebSocket	D-Bus		
Success	SUCCESS: HMI provides the information about ‘mixing audio’ supporting.	JSON response	Regular response	attenuatedSupported, code: 0	-
Failure	GENERIC_ERROR: The unknown issue occurred or other codes are not applicable.	JSON <u>error message</u>	Regular <u>response</u>	code: 22	Applicable for this RPC result codes. Please see Result Enumeration for all SDL-supported codes

6.3.3.2 Parameters

Param Name	Type	Mandatory	Description
attenuatedSupported	Boolean	true	Must be - ‘true’ if HMI supports mixing audio - ‘false’ if doesn’t.

6.3.4 Sequence Diagrams

6.3.4.1 MixingAudioSupported Messaging



6.3.5 JSON Messages Examples

6.3.5.1 Request

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

6.3.5.2 Response

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

6.3.5.3 Error message

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

```

    }
}
```

6.4 AllowDeviceToConnect

6.4.1 Description

Type:	Function
Sender:	SDL
Purpose:	Permit device to connect to HU.

6.4.2 Request

6.4.2.1 Behavior

HMI must:

- 1) Check in one of predefined ways whether to allow the named device to connect to HU.

HMI may:

- 1) Request the User's answer in one of accessible ways (VR, buttons, UI pop-up etc.)
- 2) Have the use-case to define whether to allow the connection or not

6.4.2.2 Parameters

Param Name	Type	Mandatory	Description
deviceInfo	Common.DeviceInfo	true	The information about the device requested to be connected: its name and ID. See DeviceInfo .

6.4.2.3 DeviceInfo Structure

Param Name	Type	Mandatory	Additional	Description
name	String	true		The name of the device connected
id	String	true		The ID of the device connected. either hash of device's USB serial number (in case of USB connection) or hash of device's MAC address (in case of BlueTooth or WiFi connection). It remains unique between the ignition cycles for the same transport type .
transportType	Common.TransportType	false		The transport type the named-app's-device is connected over to HU (BlueTooth, USB or WiFi). Always returned by SDL in OnAppRegistered and UpdateAppList RPCs.

Param Name	Type	Mandatory	Additional	Description
isSDLAllowed	Boolean	false		Sent by SDL in UpdateDeviceList. 'true' – if device is allowed for PolicyTable Exchange; 'false' – if device is NOT allowed for PolicyTable Exchange

6.4.2.4 TransportType Enumeration

Element name	Value	Short Description
BLUETOOTH	0	
USB_IOS	1	
USB_AOA	2	
WIFI	3	

6.4.3 Response

6.4.3.1 Behavior

Note:

There is a difference in message type for WebSocket and D-Bus connection (see the table below).

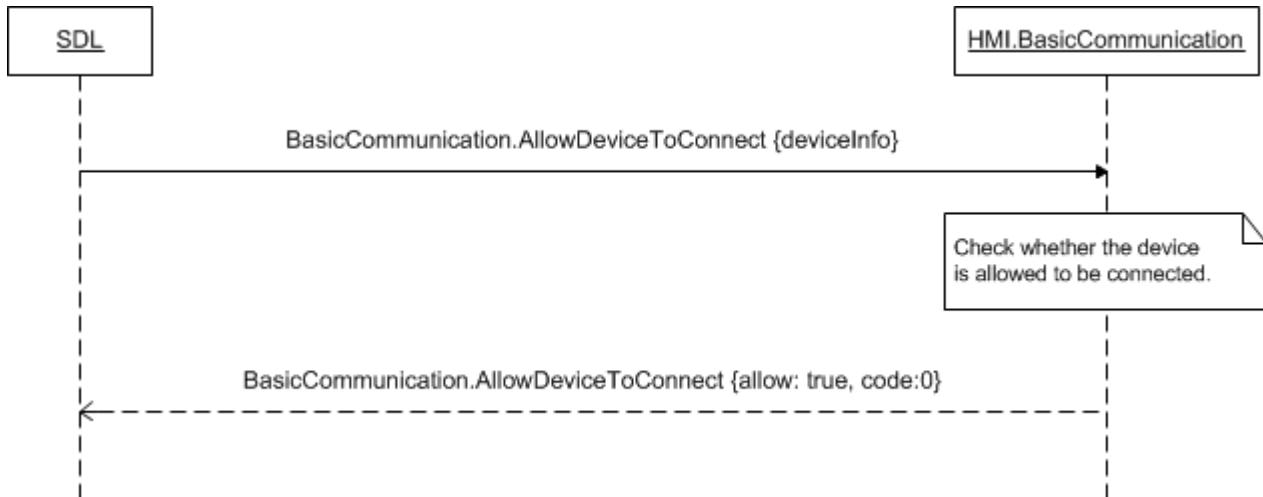
Result	Description	Message type		Message Params	Notes
		WebSocket	D-Bus		
Success	SUCCESS: HMI provides the information about whether the named device is allowed.	JSON response	Regular response	allow, code: 0	-
Failure	REJECTED A User rejected the device to be connected	JSON error message	Regular response	code: 4	
	IGNORED The permission has been already done by the user			code: 6	
	TIMED_OUT no answer from a user within predefined timeout on HMI			code: 10	
	INVALID_DATA Wrong JSON or out of bound parameters			code: 11	
	GENERIC_ERROR: The unknown issue occurred or other codes are not applicable.			code: 22	Applicable for this RPC result codes. Please see Result Enumeration for all SDL-supported codes

6.4.3.2 Parameters

Param Name	Type	Mandatory	Description
allow	Boolean	true	Must be - 'true' if connecting the requested device is allowed - 'false' if not.

6.4.4 Sequence Diagrams

6.4.4.1 AllowDeviceToConnect Messaging



6.4.5 JSON Messages Examples

6.4.5.1 Request

```
{
  "id" : 87,
  "jsonrpc" : "2.0",
  "method" :
"BasicCommunication.AllowDeviceToConnect",
  "params" :
  [
    "deviceInfo" :
    {
      "name" : "Mary`s Phone",
      "id" : 8
    }
  ]
}
```

6.4.5.2 Response

```
{
  "id" : 87,
  "jsonrpc" : "2.0",
  "result" :
  {
    "allow" : true,
    "code" : 0,
    "method" :
```

```

    "BasicCommunication.AllowDeviceToConnect"
    }
}

```

6.4.5.3 Error message

```

{
  "id" : 87,
  "jsonrpc" : "2.0",
  "error" :
  {
    "code" : 22,
    "message" : "An unknown error
occurred",
    "data" :
    {
      "method" :
      "BasicCommunication.AllowDeviceToConnect"
    }
  }
}

```

6.5 SystemRequest

6.5.1 Description

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

SDL sends BasicCommunication.SystemRequest to HMI in case SDL receives SystemRequest RPC from mobile application.

6.5.2 Request

6.5.2.1 Behavior

HMI must:

- 1) Notify specific module about the location of fully assembled packet of filename file transferred via PutFile based on RequestType allowed

SDL related behavior:

SDL validates all SystemRequests sent from mobile app and returns 'DISALLOWED' result in case app's SystemRequest contains RequestType disallowed by Policies. See [6.5.4 SystemRequest diagram for details](#).

Note: SDL sends the list of RequestTypes allowed by Policies via OnAppPermissionChanged/UpdateAppList or OnAppRegistered API. In case HMI sends OnSystemRequest with a RequestType disallowed by PolicyTable, SDL will ignore it.

6.5.2.2 Parameters

Param Name	Type	Mandatory	Additional	Description
requestType	Common.RequestType	true	-	The type of system request.
fileName	String	true	minlength = 1 maxlength = 255	The path to the file that SDL stored .in predefined in <i>smartDeviceLink.ini</i> file folder.
appID	String	true		Internal ID of the application that corresponds to the policyAppID

6.5.2.3 RequestType

Element name	Value	Short 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	
MEDIA	18	
FOTA	19	See chapter 6.19 OnSystemRequest

6.5.3 Response

Note:

There is a difference in message type for WebSocket and D-Bus connection (see the table below).

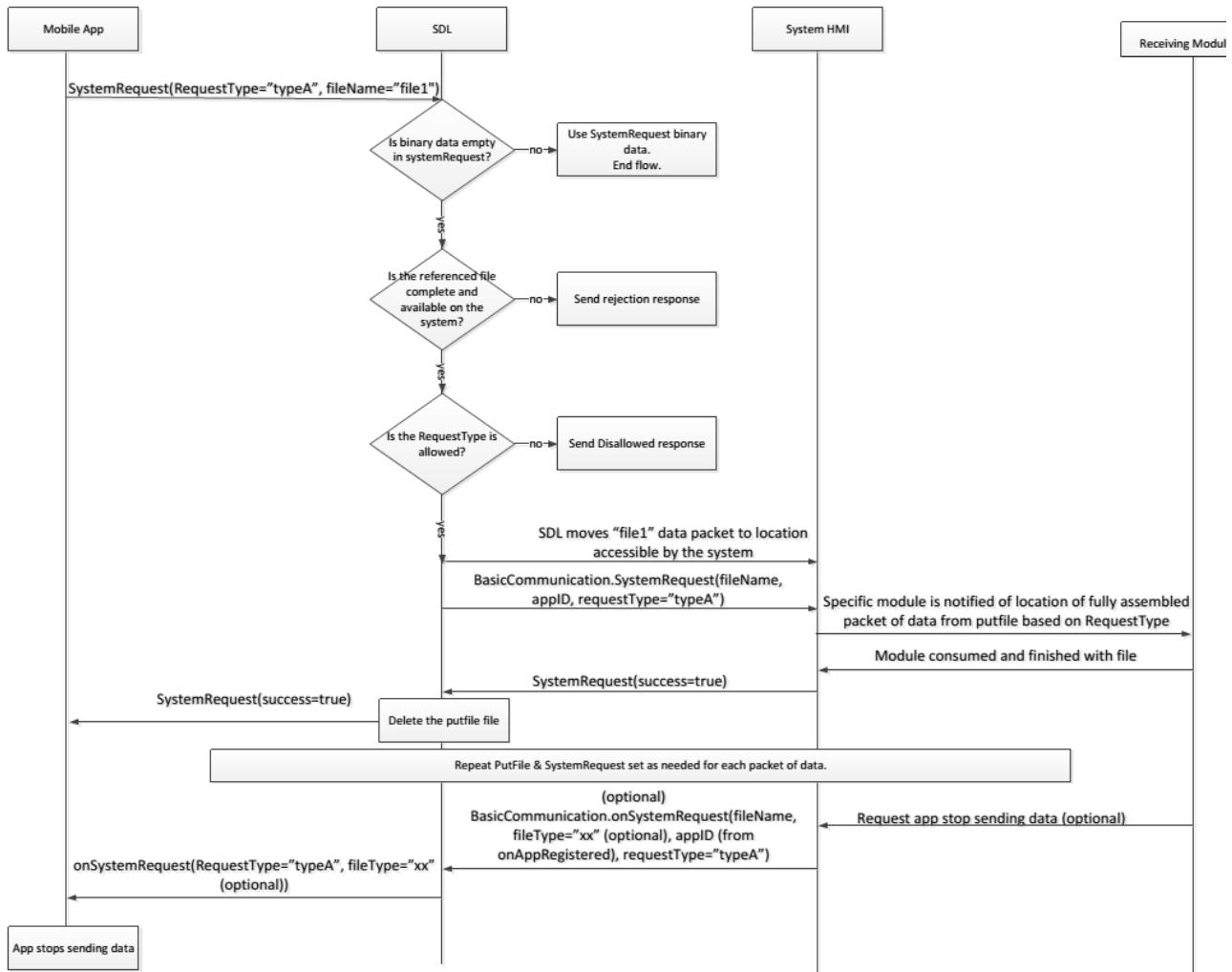
Result	Description	Message type		Message Params	Notes
		WebSocket	D-Bus		
Success	SUCCESS: HMI recognizes the path to the file and the corresponding request type..	JSON response	Regular response	code: 0	-
Failure	INVALID_DATA Wrong json format or	JSON	Regular	code: 11	

	out of bound parameters values	response	response		
	INVALID_ID Wrong appID or policyAppID			code: 13	Actually the check is performed on SDL side, but in case by any reason HMI checks the data too, the code must be return if the appID or policyAppID is invalid
	OUT_OF_MEMORY			code: 17	Actually the check is performed on SDL side, but in case by any reason HMI may check the data too, the code must be return if it's not enough memory to run the request
	GENERIC_ERROR: 1) The unknown issue occurred or other codes are not applicable.			code: 22	Applicable for this RPC result codes. Please see Result Enumeration for all SDL-supported codes

Important 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

6.5.4 Sequence Diagrams

6.5.4.1 SystemRequest workflow



6.5.5 JSON Messages Examples

6.5.5.1 Request

```
{
    "id" : 59,
    "jsonrpc" : "2.0",
    "method" :
"BasicCommunication.SystemRequest"
    "params" :
    {
        "requestType" : FILE_RESUME,
        "fileName" :
"/tmp/fs/mp/images/ivsu_cache/123.json",
        "appID" : "824587763458"
    }
}
```

6.5.5.2 Response

```
{
    "id" : 59,
```

```

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

```

6.5.5.3 Error message

```

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

```

6.6 GetSystemInfo

6.6.1 Description

Type:	Function
Sender:	SDL
Purpose:	Get information about system information of HU.

Request from SDL to HMI to obtain information about head unit system main parameters. This information is used by PolicyManager for updating the policies database (see [6.7.4.1 GetSystemInfo](#))

6.6.2 Request

6.6.2.1 Behavior

- 1) **HMI must:** Respond on BC.GetSystemInfo and provide the current values of software version of HU, language and WERS country code.

SDL Note: When SDL has been just started, it sends GetSystemInfo on each start up to update the policies database with system data. It stores the "ccpu" version to respond to application's RegisterAppInterface request further.

6.6.3 Response

Note:

There is a difference in message type for WebSocket and D-Bus connection (see the table below).

Result	Description	Message type		Message Params	Notes
		WebSoc ket	D-Bus		

Success	SUCCESS The information is available and provided.	JSON response	Regular response	allow, Code: 0	See section 6.6.3.2
Failure	DATA_NOT_AVAILABLE Requested data or a part of the requested data isn't available	JSON error message	Regular response	parameters which are available, code:9	
	TIMED_OUT No response from the system within system defined timeout			code: 10	
	GENERIC_ERROR The information is not available or some failure occurred.			Code: 22	Result code must correspond to the failure occurred.

6.6.3.1 Parameters

Param Name	Type	Mandatory	Additional	Description
ccpu_version	String	true	maxlength = 500	Software version of the module
language	Common.Language	true	-	ISO 639-1 combined with ISO 3166 alpha-2 country code (i.e. en-us)
wersCountryCode	String	true	maxlength = 500	Country code from the Ford system WERS (i.e.WAEGB).

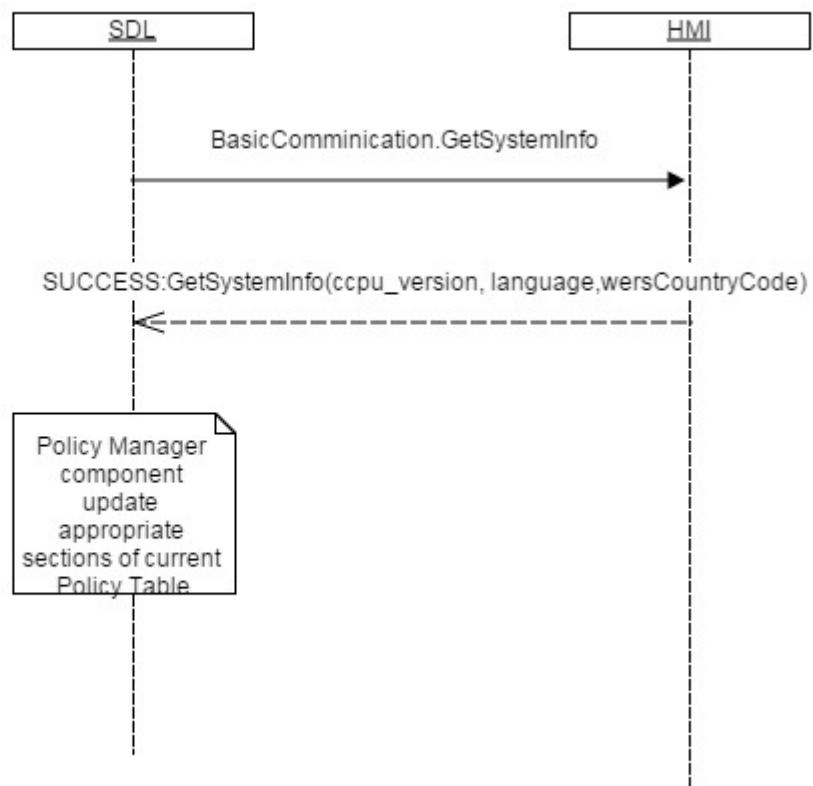
6.6.3.2 Language Enumeration

Element name	Value	Short Description
AR-SA	0	Arabic – Saudi Arabia
CS-CZ	1	Czech – Czech Republic
DA-DK	2	Danish – Denmark
DE-DE	3	German – Germany
EN-AU	4	English – Australia
EN-GB	5	English – GB
EN-US	6	English – US
ES-ES	7	Spanish – Spain
ES-MX	8	Spanish – Mexico
FR-CA	9	French – Canada
FR-FR	10	French – France
IT-IT	11	Italian – Italy
JA-JP	12	Japanese – Japan
KO-KR	13	Korean – South Korea
NL-NL	14	Dutch (Standard) – Netherlands
NO-NO	15	Norwegian - Norway
PL-PL	16	Polish – Poland
PT-PT	17	Portuguese – Portugal
PT-BR	18	Portuguese – Brazil
RU-RU	19	Russian - Russia

Element name	Value	Short Description
SV-SE	20	Swedish – Sweden
TR-TR	21	Turkish – Turkey
ZH-CN	22	Mandarin – China
ZH-TW	23	Mandarin – Taiwan
NL-BE	24	Dutch Belgium (Flemish)
EL-GR	25	Greek
HU-HU	26	Hungarian
FI-FI	27	Finnish
SK-SK	28	Slovak

6.6.4 Sequence Diagrams

6.6.4.1 GetSystemInfo



6.6.5 JSON Messages Examples

6.6.5.1 Request

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

6.6.5.2 Response

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

6.6.5.3 Error message

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

6.7 OnReady

6.7.1 Description

Type:	Notification
Sender:	HMI
Purpose:	Inform about readiness to communicate.

This is the first message in SDL-HMI communication.

Note: SDL ignores all invalid notifications which come from HMI (invalid JSON, invalid data types/bounds etc)

HMI must:

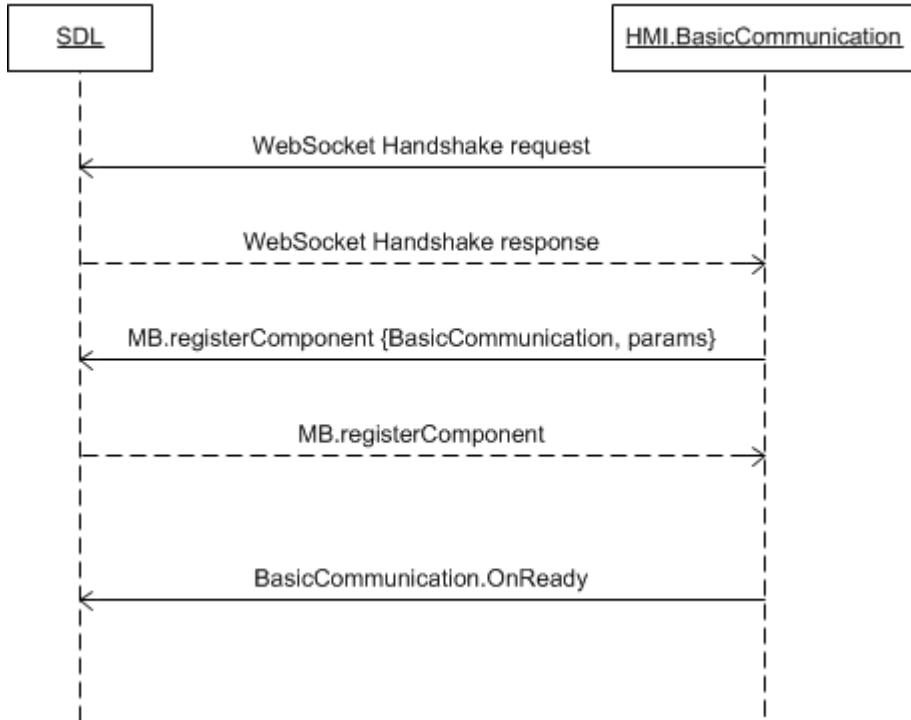
Send this notification after the connection is established and HMI is ready for communication:

- 1) For WebSocket connection: right after the handshake for HMI's BasicCommunication component and its registering with SDL (see [section 2.1.2](#) and [section 2.1.3](#)).

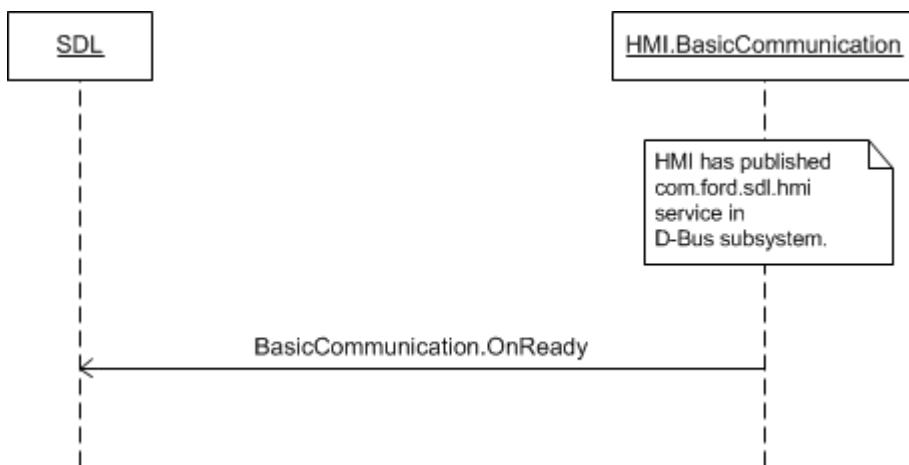
- 2) For D-Bus connection: right after the service `com.ford.sdl.hmi` is published.

6.7.2 Sequence Diagrams

6.7.2.1 *OnReady after WebSocket connection establishment*



6.7.2.2 *OnReady after D-Bus service publishing*



6.7.3 JSON Messages Examples

```
{
  "jsonrpc" : "2.0",
  "method" :
```

```
"BasicCommunication.OnReady"  
}
```

6.8 OnStartDeviceDiscovery

6.8.1 Description

Type:	Notification
Sender:	HMI
Purpose:	Initiate device search.

On receipt of this notification SDL starts searching for devices on all of available transports. Afterwards, SDL provides the search results via [UpdateDeviceList](#) RPC.

Note: SDL ignores all invalid notifications which come from HMI (invalid JSON, invalid data types/bounds etc)

HMI must:

- 1) Provide the User with the possibility to choose the device discovery in one of accessible ways (VR, buttons, pop-ups etc.).
- 2) Send `OnStartDeviceDiscovery` notification when the User has chosen to search for devices.

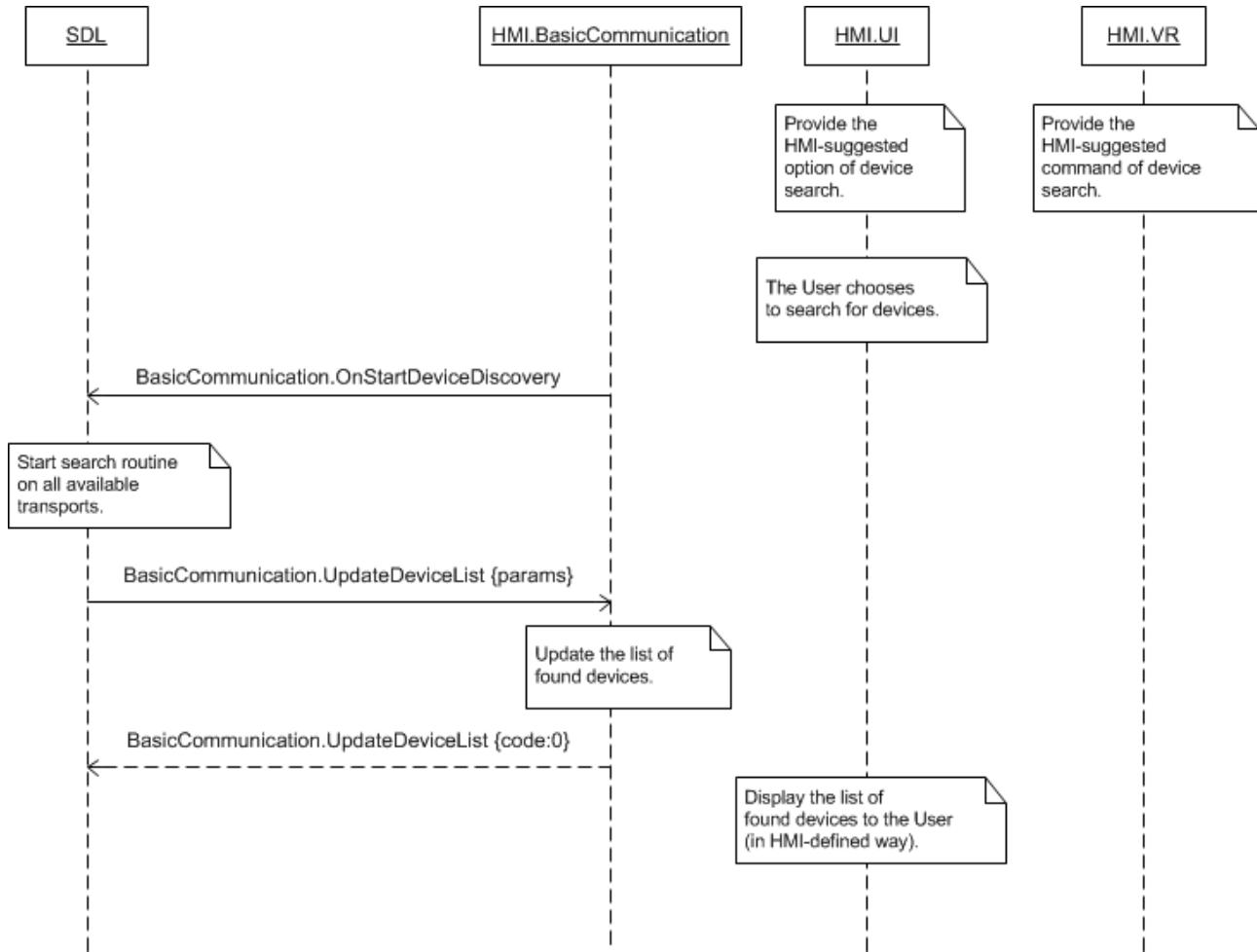
Note:

The difference:

- *OnStartDeviceDiscovery initiates SDL to start the new search procedure*
- *OnUpdateDeviceList results SDL to provide the updates of the recent search.*

6.8.2 Sequence Diagrams

6.8.2.1 *OnStartDeviceDiscovery upon User's request and resulting UpdateDeviceList*



6.8.3 JSON Messages Examples

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

6.9 OnDeviceChosen

6.9.1 Description

Type:	Notification
Sender:	HMI
Purpose:	Open the connection with the named device.

On receipt of this notification SDL opens the connection with the named device and starts session with the application(s) running on it.

HMI must:

- 1) Provide the User with the possibility to choose among the list of found devices in one of accessible ways (VR, buttons, etc.).
- 2) Send `OnDeviceChosen` notification when the User has chosen the device.

Notes:

The list of found devices (that HMI may provide to the User for choosing) is provided by SDL within [UpdateDeviceList](#) request.

SDL ignores all invalid notifications which come from HMI (invalid JSON, invalid data types/bounds etc)

6.9.1.1 Parameters

Param Name	Type	Mandatory	Description
deviceInfo	Common.DeviceInfo	true	The information about the device chosen by the User: its name and ID. See <code>DeviceInfo</code> .

6.9.1.2 DeviceInfo Structure

Param Name	Type	Mandatory	Additional	Description
name	String	true		The name of the device connected
id	String	true		The ID of the device connected. either hash of device's USB serial number (in case of USB connection) or hash of device's MAC address (in case of BlueTooth or WiFi connection). It remains unique between the ignition cycles for the same transport type .
transportType	Common.TransportType	false		The transport type the named-app's-device is connected over to HU (BlueTooth, USB or WiFi). Always returned by SDL via <code>OnAppRegistered</code> and <code>UpdateAppList</code> RPCs.
isSDLAllowed	Boolean	false		Sent by SDL in <code>UpdateDeviceList</code> . 'true' – if device is allowed for <code>PolicyTable Exchange</code> ; 'false' – if device is NOT allowed for <code>PolicyTable Exchange</code>

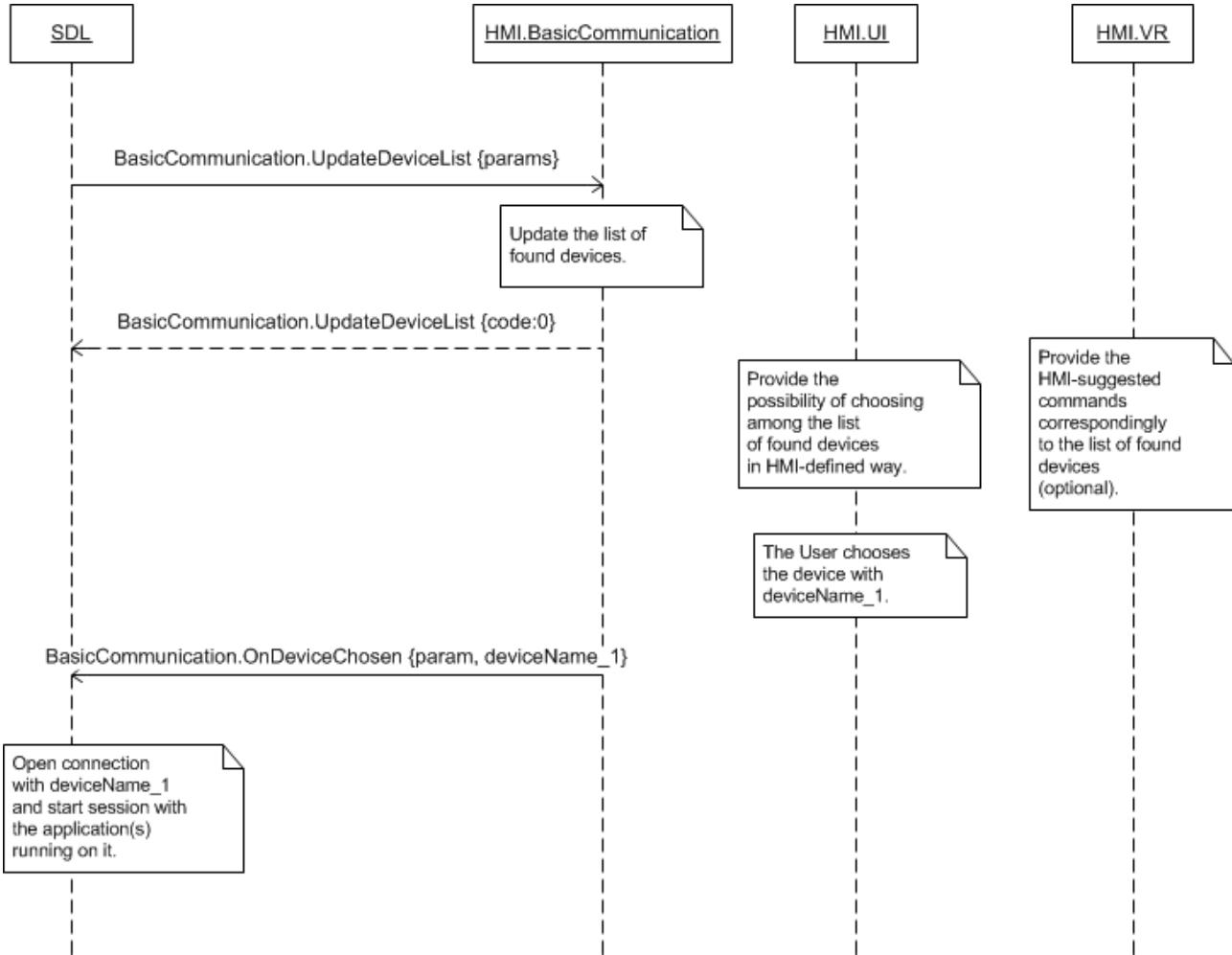
6.9.1.3 TransportType Enumeration

Element name	Value	Short Description
BLUETOOTH	0	

Element name	Value	Short Description
USB_IOS	1	
USB_AOA	2	
WIFI	3	

6.9.2 Sequence Diagrams

6.9.2.1 OnDeviceChosen with preceding UpdateDeviceList (BT transport)



6.9.3 JSON Messages Examples

```
{
  "jsonrpc" : "2.0",
  "method" :
"BasicCommunication.OnDeviceChosen",
  "params" :
{
  "DeviceInfo" :
{
    "name" : "Jerry's Phone",
    "id" : 3
  }
}
```

```
}
```

6.10 OnFindApplications

6.10.1 Description

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

On receipt of this notification SDL provides the list of registered applications for the named device via [UpdateAppList](#) RPC.

Note: SDL ignores all invalid notifications which come from HMI (invalid JSON, invalid data types/bounds etc)

HMI must:

- 1) Send OnFindApplications notification upon User's request to find the apps on a device (e.g. after choosing the device in the list of found devices or on definite button click, depends on HMI workflow)

6.10.1.1 Parameters

Param Name	Type	Mandatory	Description
deviceInfo	Common.DeviceInfo	true	The information about the device chosen by the User: its name and ID. See DeviceInfo .

6.10.1.2 DeviceInfo Structure

Param Name	Type	Mandatory	Additional	Description
name	String	true		The name of the device connected
id	String	true		The ID of the device connected. either hash of device's USB serial number (in case of USB connection) or hash of device's MAC address (in case of BlueTooth or WiFi connection). It remains unique between the ignition cycles for the same transport type .
transportType	Common.TransportType	false		The transport type the named-app's-device is connected over to HU (BlueTooth, USB or WiFi). Always returned by SDL in OnAppRegistered and UpdateAppList RPCs.

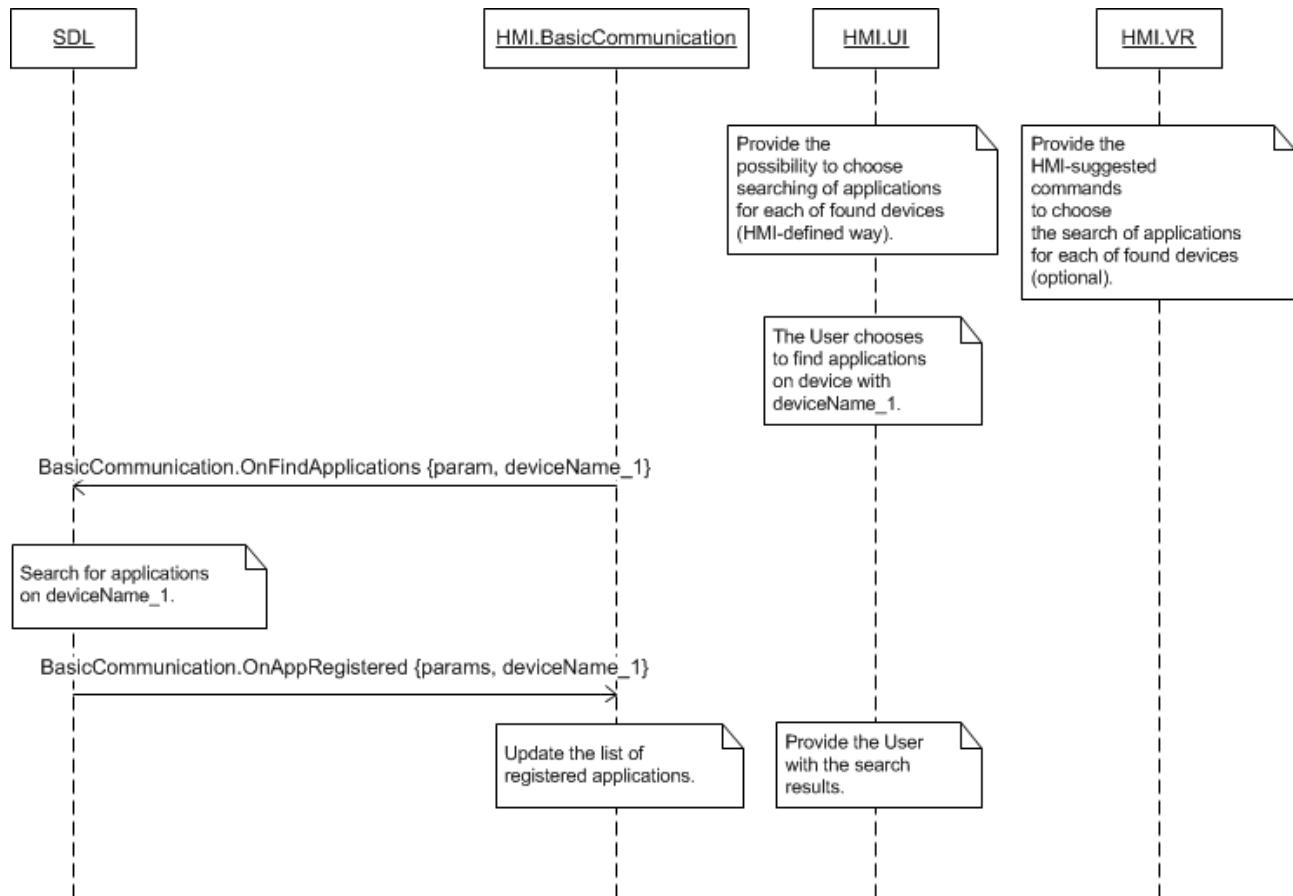
Param Name	Type	Mandatory	Additional	Description
isSDLAllowed	Boolean	false		Sent by SDL in UpdateDeviceList. 'true' – if device is allowed for PolicyTable Exchange; 'false' – if device is NOT allowed for PolicyTable Exchange

6.10.1.3 TransportType Enumeration

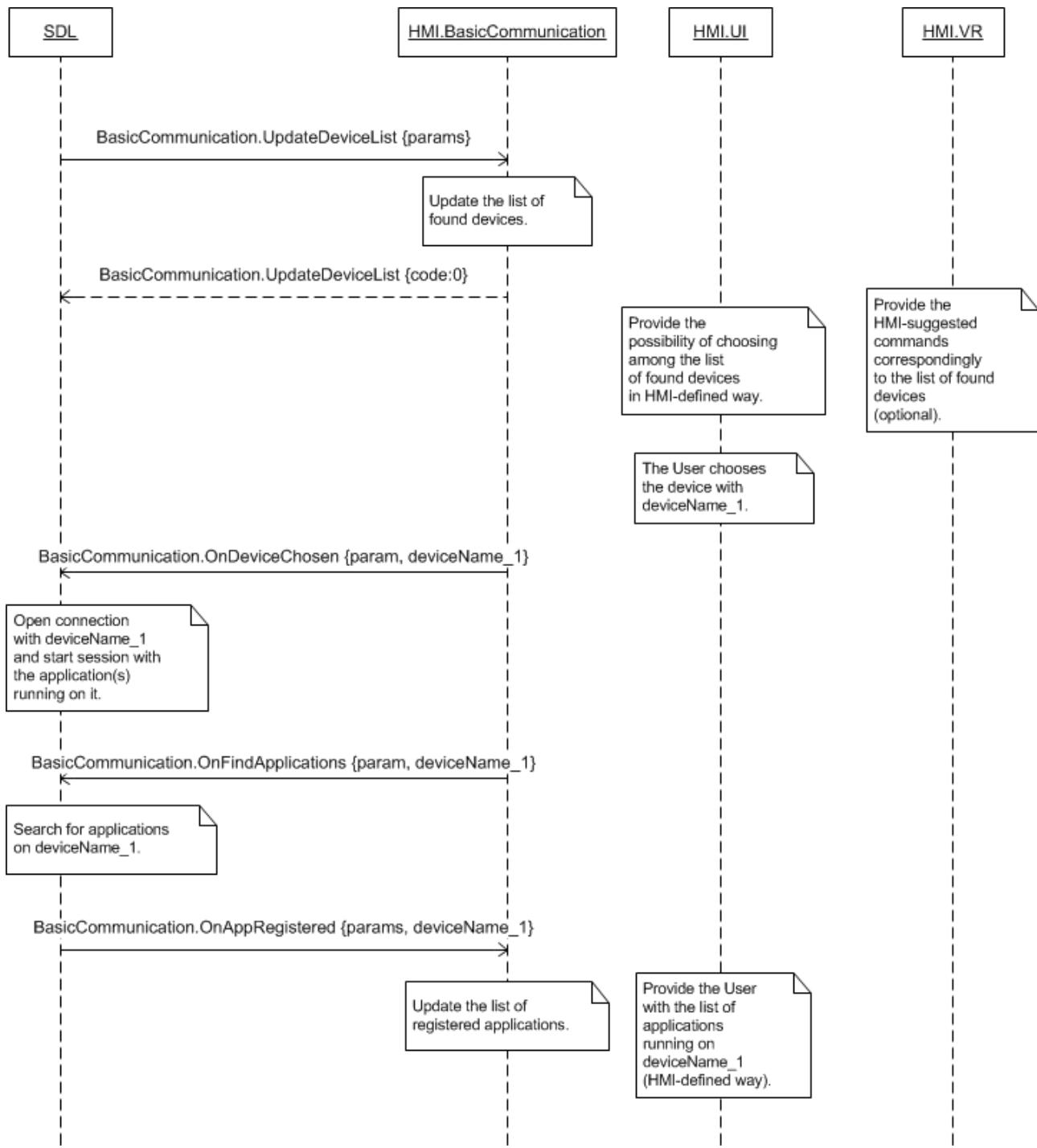
Element name	Value	Short Description
BLUETOOTH	0	
USB_IOS	1	
USB_AOA	2	
WIFI	3	

6.10.2 Sequence Diagrams

6.10.2.1 OnFindApplications upon User's choice and resulting OnAppRegistered



6.10.2.2 OnFindApplications upon User's choosing the device in the list of found devices and resulting OnAppRegistered



6.10.3 JSON Messages Examples

```
{
    "jsonrpc" : "2.0",
    "method" :
    "BasicCommunication.OnFindApplications",
    "params" :
    {
```

```

"deviceinfo" :
{
    "name" : "XT910",
    "id" : 4
}
}

```

6.11 OnAppActivated

6.11.1 Description

Type:	Notification
Sender:	HMI
Purpose:	Inform about the User has chosen to activate the application.

SDL requires this notification to know the user requests application to be in focus and start operate with related to HMI's functionality

On coming of `OnAppActivated` notification SDL provides [ActivateApp](#) RPC confirming the named application may be activated.

HMI must:

- 1) Send `OnAppActivated` notification when the User chooses (in one of HMI-suggested ways) the application on HMI.
- 2) Wait for [ActivateApp](#) RPC and only then display the named application related screen.

Notes:

The information about the application (name, ID, etc.) is provided by SDL via either UpdateAppList or OnAppRegistered RPCs.

SDL ignores all invalid notifications which come from HMI (invalid JSON, invalid data types/bounds etc)

Note:

The difference:

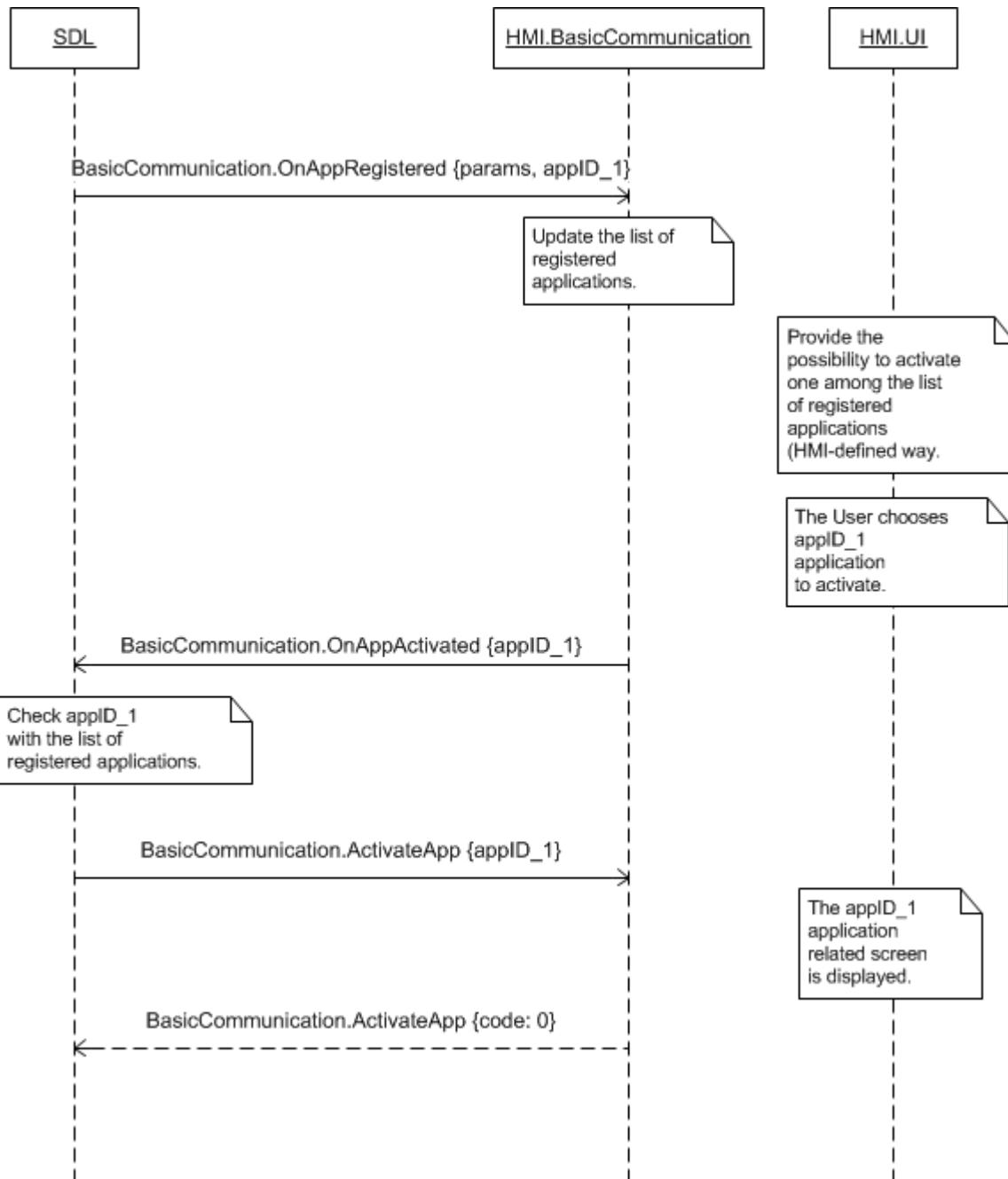
- `OnAppActivated` shows User's intension to activate the named application.
- `ActivateApp` is SDL's request and permission for the named application to be granted with the access to HMI's functionality.

6.11.1.1 Parameters

Param Name	Type	Mandatory	Description
appID	Integer	true	ID of the application to be activated on HMI.

6.11.2 Sequence Diagrams

6.11.2.1 OnAppActivated upon User's choice and resulting ActivateApp



6.11.3 JSON Messages Examples

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

}

6.12 OnAppDeactivated

6.12.1 Description

Type:	Notification
Sender:	HMI
Purpose:	Inform about the application is no longer active on HMI.

SDL requires this notification for restricting the application on mobile device from sending the RPCs related to HMI's functionality (e.g. adding commands for VR, starting interaction with the User, speaking the text via TTS, etc.)..

HMI must:

- 1) Send OnAppDeactivated notification when the User has navigated away from the application persistent screen because of some DeactivateReason (see [6.13.1.2 DeactivateReason Enumeration](#))
- 2) Send OnAppDeactivated (reason= "NAVIGATION", appId) from HMI when the user switches from the application layout to embedded Navigation screen
- 3) Send OnAppDeactivated (reason= "AUDIO") from HMI when the user switches to embedded Audio source
- 4) Send OnAppDeactivated (reason= "PHONE") from HMI when the user switches to HMI phone menu

SDL information:

- 1) In case Navigation application with appId is in *FULL* or *LIMITED* and HMI sends OnAppDeactivated (reason= "NAVIGATIONMAP", appId), SDL **must** put Navigation application to *BACKGROUND*
- 2) In case Navigation application application with appId in *FULL* or *LIMITED* and HMI sends OnAppDeactivated(DeactivateReason "AUDIO", appId), SDL **must** put Navigation app to *LIMITED* HMI level and *AUDIBLE* *AudioStreamingState*.

Note:

The information about the application (name, ID, etc.) is provided by SDL via either UpdateAppList or OnAppRegistered RPCs.

SDL ignores all invalid notifications which come from HMI (invalid JSON, invalid data types/bounds etc)

6.12.1.1 Parameters

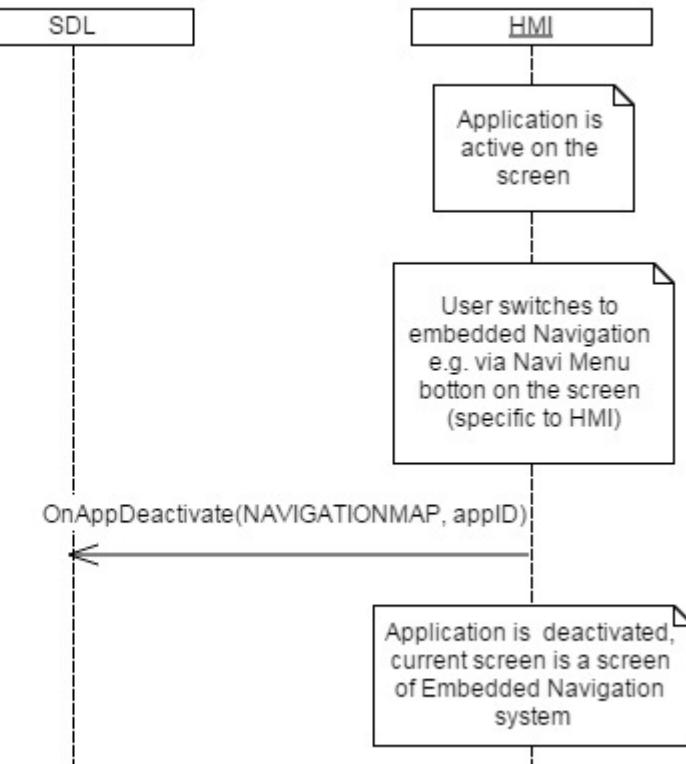
Param Name	Type	Mandatory	Description
appID	Integer	true	ID of the application deactivated.
reason	Common.DeactivateReason	true	Specifies the functionality the User has switched to. See <code>DeactivateReason</code> .

6.12.1.2 DeactivateReason Enumeration

Element name	Short Description
AUDIO	The User has navigated to embedded audio player (radio etc.)
PHONECALL	Active phone call deactivated the application (see also OnPhoneCall)
NAVIGATIONMAP	The User has navigated to embedded navigation system screen
PHONEMENU	The User has navigated to phone menu.
SYNCSETTINGS	The User has navigated to HU settings menu.
GENERAL	Other screens navigation apart from other mobile app.

6.12.2 Sequence Diagrams

6.12.2.1 OnAppDeactivated Simple Messaging Example



See also sample of using the notification in [6.16.2.1 OnExitApplication USER_EXIT](#).

6.12.3 JSON Messages Examples

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

6.13 OnAppRegistered

6.13.1 Description

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

SDL may send the notification:

- After SDL-enabled application has registered itself on SDL successfully (either for the very 1st time or registered in previous ignition cycles).
- When the device was re-connected to HU after unexpected disconnect/ and previously registered SDL-enabled application has re-registered on SDL successfully. SDL makes decision if data resumption process is applicable for the application
- Upon User's request delivered by HMI via `OnFindApplications` notification, in case any new applications are registered.

Note: SDL sends the list of RequestTypes allowed by Policies via `OnAppRegistered` API. In case HMI sends `OnSystemRequest` with a RequestType disallowed by PolicyTable, SDL will ignore it).

Note about Data resumption:

Data resumption means that app may *request* to restore data operated in the previous ign cycle(s)/after unexpected disconnect. .

- *For data resumption purposes SDL must keep storing such application-related data like commands, application global properties and show data during three ignition cycles after unexpected disconnect or ignition off. Upon the forth "Ignition On" SDL clears all corresponding application-related data for resumption.*

- HMI must store only Vrgrammars compiled for the applications unregistered by unexpected disconnect or Ignition Off

If the application resumes data successfully

- SDL will provide OnAppRegistered with resumeVrGrammars : true to notify HMI about VRGrammars must be resumed
- SDL must restore application-related data:
 - AddCommand (Menu +VR)
 - AddSubMenu
 - CreateInteractionChoiceSet
 - SetGlobalProperties
 - SubscribeButton
 - SubscribeVehicleData
- SDL must send the following restored data to HMI right after OnAppRegistered notification sent:
 - AddCommand (Menu only)
 - AddSubMenu
 - CreateInteractionChoiceSet
 - SetGlobalProperties
 - SubscribeButton
 - SubscribeVehicleData

- HMI is notified about application resumption case via OnAppRegistered with resumeVrGrammars : true . On this event, HMI must restore the application related VRGrammars previously stored for the application with appID received in OnAppRegistered notification

If the application does NOT resume data successfully:

- SDL will provide OnAppRegistered with resumeVrGrammars : false or no resume parameter
- SDL cleans up all previously stored application data for the application that failed to resume its dataHMI must clean up previously compiled VRGrammars for the application with appID received via OnAppRegistered notification

After failed data resumption, application will send new data required for its operation through SDL to HMI. In this case a new cycle of collecting app's data for resumption must begin on both SDL and HMI sides

Important:

When the application is registering after unexpected disconnect or Ignition Off, not only the application data but Active Source may be resumed on HMI. These data and audio source resumption processes are independent to each other. To review the cases of AudioSource resumption refer to [BC.OnResume AudioSource](#) and [BC.OnAppActivated](#) chapters.

HMI must:

- 1) Update its list of registered applications.
- 2) Store the application data sent with `applications` parameter (application name, icon, name and ID of corresponding device, `appID` and etc. that is described in section [6.13.1.2 HMIApplication](#)).

Note: The value of `appID` provided by SDL will be the same between registrations only in case of "resumeVrGrammers:true". Otherwise, SDL provides different `appIDs` for one and the same application between the registration- Remember the value of `ttsName` parameter and speak this information (which is the application name) when the User speaks 'Help' in the VR layer that corresponds 'the list of registered applications' on UI.

- Compile and store VRGrammars for `vrSynonyms` parameter and use them when arranging the possibility for the User to choose among the registered applications by VR. The commands must be accessible on the layer correspondent to the following entries on UI:

- The list of registered applications
- Any active application (that is, HMI must provide the possibility to activate any of registered applications by VR if another application is currently active).

- Provide the User with the possibility to choose among registered applications on UI.

Note:

- 1) *HMI must provide `OnAppActivated` with the corresponding `appID` whenever the User activates the application from UI or via VR.*
- 2) Use `appID` provided via notification when use other API (e.g. `OnAppActivated`, `OnCommand` and other) related to the corresponding application on SDL.

SDL Note:

- 1) When device is connected over USB(AOA or iOS),and the app from this device was successfully registered, SDL must send `OnAppRegistered` with (`deviceID= hash of <usb_serial>`) notification to HMI to be able to identify the application between ignition cycles . See diagram [6.13.2.3 OnAppRegistered USB transport connection](#)
- 2) In case device was connected over Wi-Fi or BlueTooth and app from this device was successfully registered SDL must send `OnAppRegistered` with (`deviceID= hash of <mac_address>`) notification to HMI to be able to identify the application between ignition cycles. See diagram [6.13.2.4 OnAppRegistered Bluetooth transport connection](#)

6.13.1.1 Parameters

Param Name	Type	Mandatory	Additional	Description
application	Common.HMI Application	true	-	The parameter contains the detailed information about each application: its name, ID, type and other. See HMIApplication . Note: the value of appID for one and the same application most often varies between ignition cycles but is obligatory the same within reconnections.
ttsName	Common.TTS Chunk	false	Array = true Minsize = 1 Maxsize = 100	App's name for Text to Speech system. HMI must speak app's name when the User speaks 'Help' to VR module
vrSynonyms	String	false	Array = true Minsize = 1 Maxsize = 100 Maxlength = 40	Voice recognition commands that HMI must use when arranging the possibility for the User to choose among the registered applications by VR. The commands must be accessible on the layer correspondent to: - UI: 'list of registered applications' - UI: any active application (that is, HMI must provide the possibility to activate any of registered applications by VR when whichever another application is currently active). When any of provided commands is recognized (that is, application is chosen by VR), HMI must send OnAppActivated with the corresponding appID to SDL.
resumeVrGrammars	Boolean	false	-	Informs whether the application-related VR grammars must be restored.
priority	Common.App Priority	false	-	Send to HMI so that it can coordinate order of requests/notifications

6.13.1.2 HMIApplication Structure

Param Name	Type	Mandatory	Additional	Description
appName	String	true	Maxlength = 100	The mobile application name. It is unique over all applications.
ngnMediaScreenAppName	String	false	Maxlength = 100	Provides an abbreviated version of the application name (may be displayed on the NGN media screen). If not provided, the appName should be used instead (and may be truncated if too long)

Param Name	Type	Mandatory	Additional	Description
icon	String	false	–	Path to the application icon stored on HU hard disc.
deviceInfo	Common.DeviceInfo	true	–	The ID, serial number, transport type the named-app's-device is connected over to HU.
policyAppID	String	true	minlength="1" maxlength="50"	Policy appID the ID the app is registering with and recognized by policy table
appID	Integer	true	–	The application ID that remains unique during the ignition cycle. This ID will be sent by SDL further and must be provided by HMI within all the RPCs related to this application.
hmiDisplayLanguageDesired	Common.Language	false	–	The language that the application intends to use. See Language.
isMediaApplication	Boolean	false	–	Indicates whether the application is a media or a non-media one. Only media applications are allowed by SDL to stream audio to HU that is audible outside of the BT media source.
appType	Common.AppHMIType	false	array = true Minsize = 1 maxsize = 100	The HMI may use this information for determining what functionality should be available for the application (e.g. NAVIGATION type of application will require displaying the information and not playing the audio). See AppHMIType.
requestType	Common.RequestType	false	Minsize = 0 Maxsize =100 array="true"	The list of SystemRequest's RequestTypes allowed by policies for the named application (The app's SystemRequest sent with RequestType out of this list will get 'disallowed' response from SDL). If SDL sends an empty array - any RequestType is allowed for this app. If SDL omits this parameter - none RequestType is allowed for this app (either this is a pre-registered app or such is dictated by policies).

6.13.1.3 AppHMIType Enumeration

Element name	Short Description
---------------------	--------------------------

Element name	Short Description
DEFAULT	The application of default type.
COMMUNICATION	The application for communication type
MEDIA	The media application
MESSAGING	The application of messaging type
NAVIGATION	The application of navigation type
INFORMATION	The application of information type
SOCIAL	The application of social type
BACKGROUND_PROCESS	The application does not require displaying the information
TESTING	The application of testing type
SYSTEM	The application of system type

6.13.1.4 AppPriority Enumeration

Element name	Value	Short Description
EMERGENCY	0	
NAVIGATION	1	
VOICE_COMMUNICATION	2	
COMMUNICATION	3	
NORMAL	4	
NONE	5	

6.13.1.5 RequestType

Element name	Value	Short 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	

Element name	Value	Short Description
EMERGENCY	17	
MEDIA	18	
FOTA	19	

6.13.1.6 DeviceInfo Structure

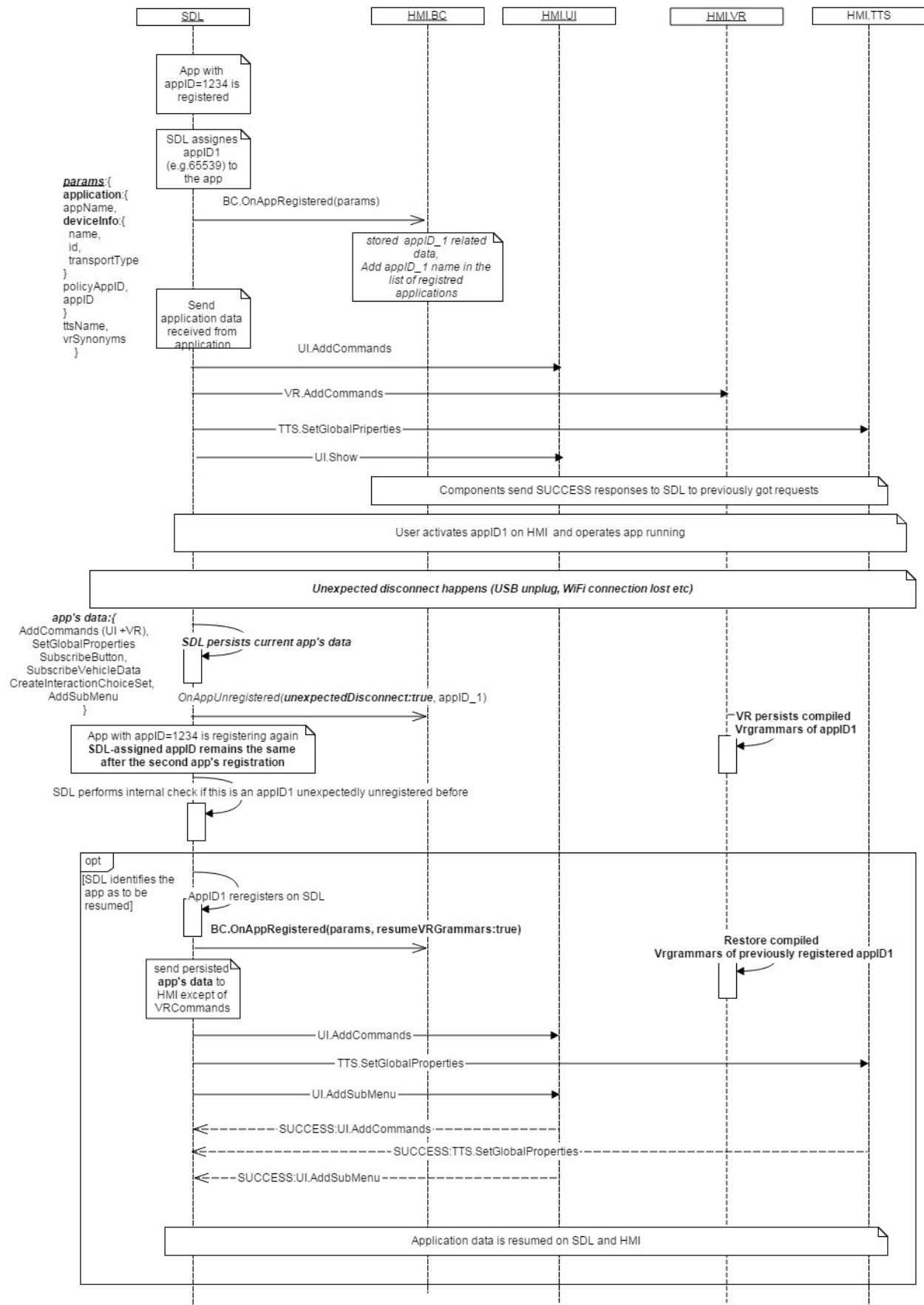
Param Name	Type	Mandatory	Additional	Description
name	String	true		The name of the device connected
id	String	true		The ID of the device connected. either hash of device's USB serial number (in case of USB connection) or hash of device's MAC address (in case of BlueTooth or WiFi connection). It remains unique between the ignition cycles for the same transport type .
transportType	Common.TransportType	false		The transport type the named-app's-device is connected over to HU (BlueTooth, USB or WiFi). Always returned by SDL in OnAppRegistered and UpdateAppList RPCs.
isSDLAllowed	Boolean	false		Sent by SDL in UpdateDeviceList. 'true' – if device is allowed for PolicyTable Exchange; 'false' – if device is NOT allowed for PolicyTable Exchange

6.13.1.7 TransportType Enumeration

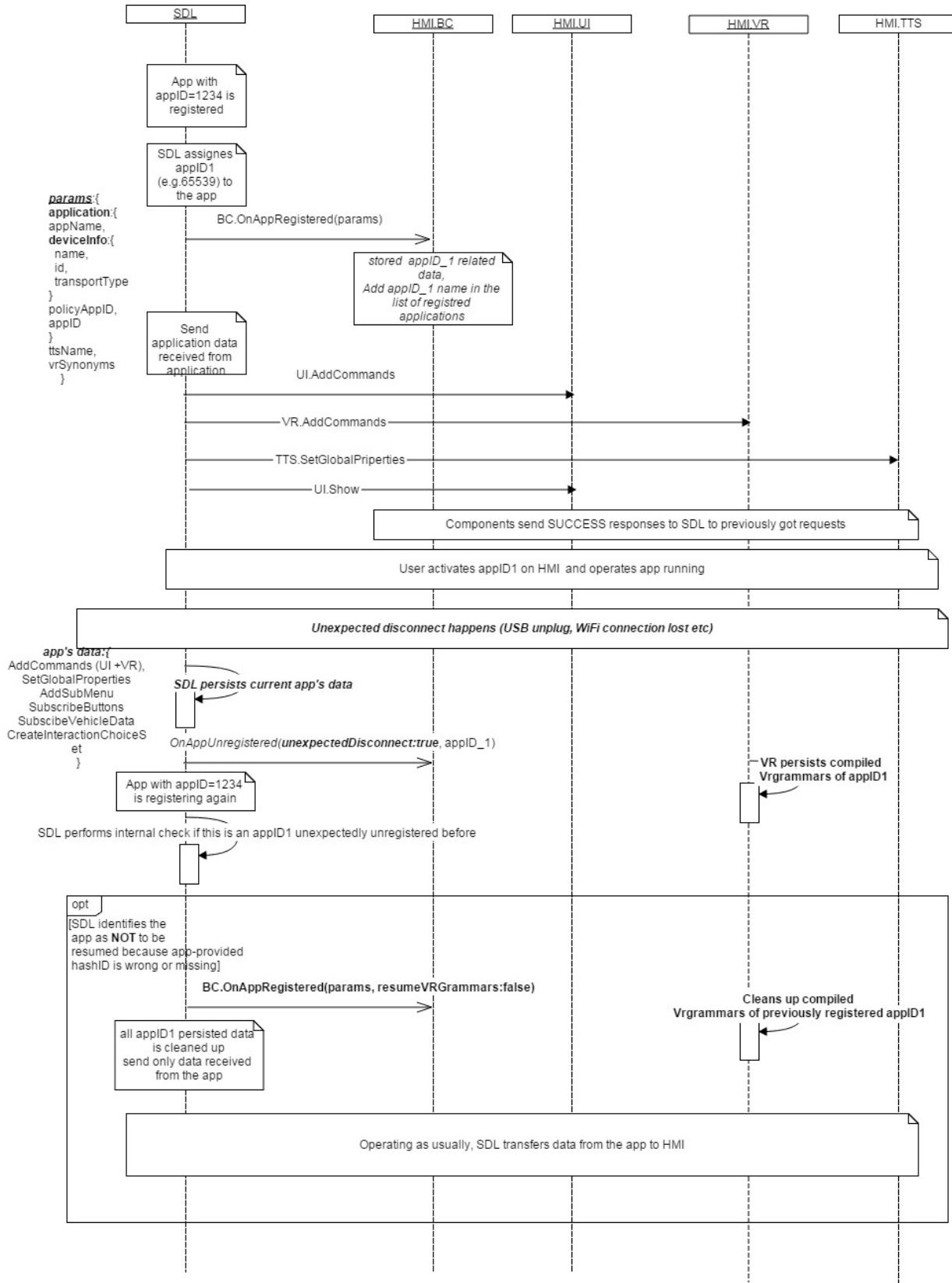
Element name	Value	Short Description
BLUETOOTH	0	
USB_IOS	1	
USB_AOA	2	
WIFI	3	

6.13.2 Sequence Diagrams

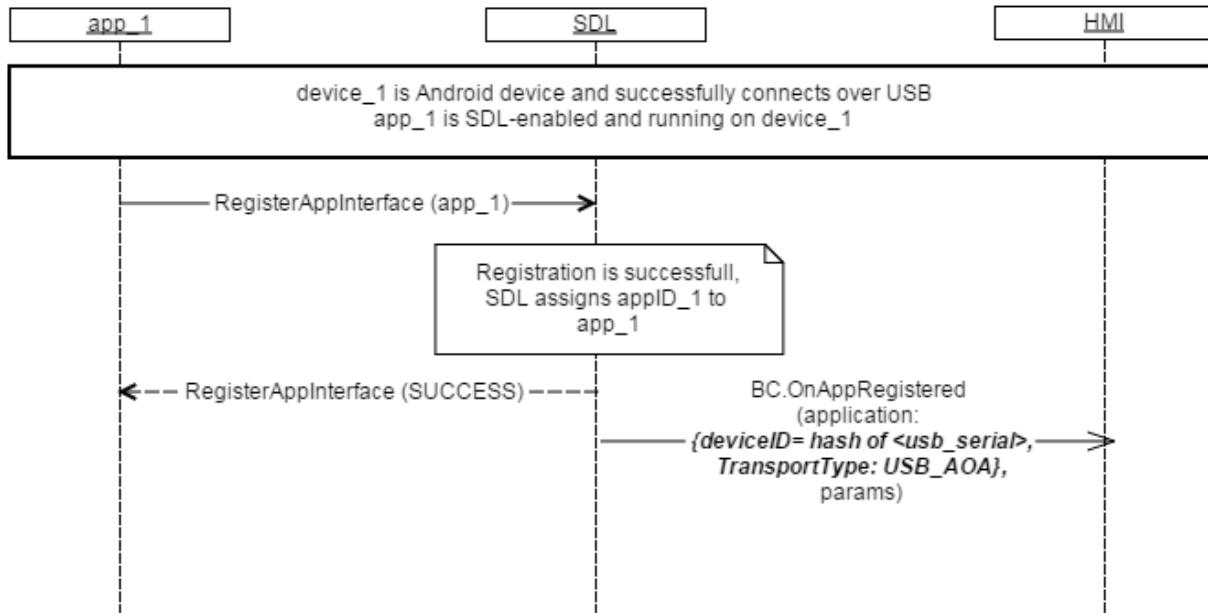
**6.13.2.1 OnAppRegistered with resume=true
after unexpected disconnect**



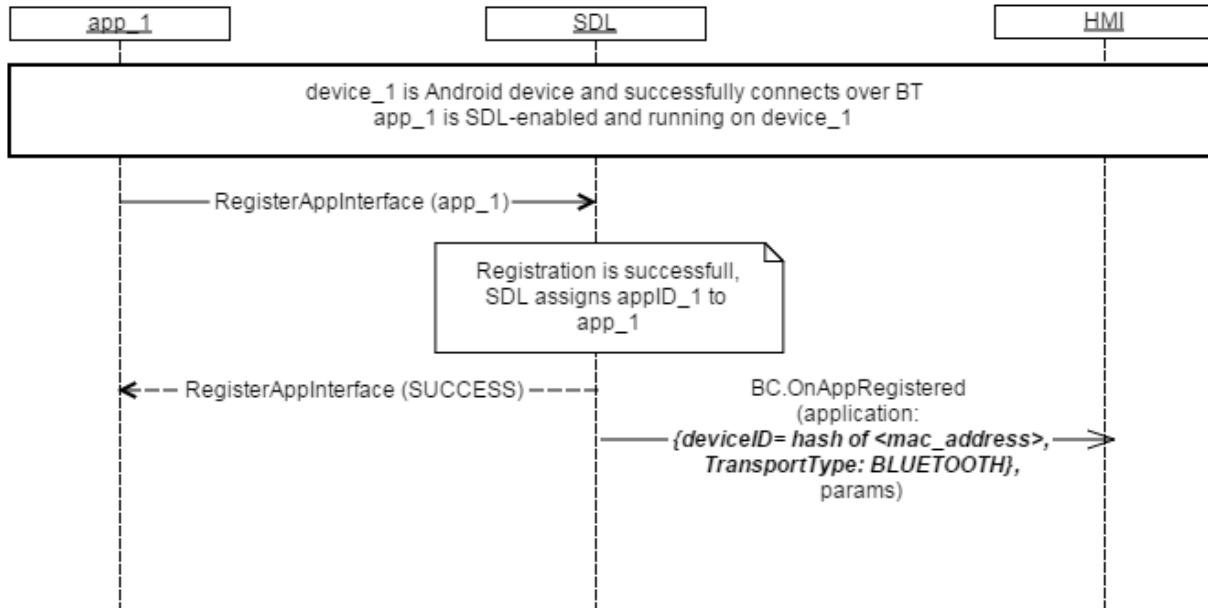
6.13.2.2 OnAppRegistered with 'resume'=false parameter after unexpected disconnect



6.13.2.3 OnAppRegistered USB transport connection



6.13.2.4 OnAppRegistered Bluetooth transport connection



6.13.3 JSON Messages Examples

```
{
```

```

    "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
}
}

```

6.14 OnAppUnregistered

6.14.1 Description

Type:	Notification
Sender:	SDL
Purpose:	Inform about the application has unregistered.

SDL sends this notification when

1. The named application has unregistered from the side of mobile device via appropriate RPC
2. The connection and the corresponding session(s) are closed due to transport issues (for example, WiFi/BT connection closing on mobile device, USB unplugging, etc.). The HeartBeat Timeout between mobile application and SDL occurs (one of SDL or application does not respond with HeartBeat message during timeout defined in *smartDeviceLink.ini* file, thus the session is closed by the responsible side).

HMI must:

- 1) Distinguish 'unexpected disconnect' and 'regular exit' events by "unexpectedDisconnect" parameter:
 - a. In case of "**unexpectedDisconnect: true**" arrives for the active app: HMI must
 - a) Display the informational HMI-defined popup of the kind: "The connection with "%AppName%" application has been unexpectedly lost".

- b) Switch the display to HMI-defined screen/page (e.g. 'Home' page, screen showing the list of registered applications, etc.).
- b. In case "**unexpectedDisconnect: false**":
HMI must proceed with step 2.
- 2) Switch the display to HMI-defined screen/page (e.g. 'Home' page, screen showing the list of registered applications, etc.) if the named application is active when this notification comes.
- 3) Remove the named application from the list of registered applications.
- 4) Delete all the data related to this application (except of VRGrammars).

Note:

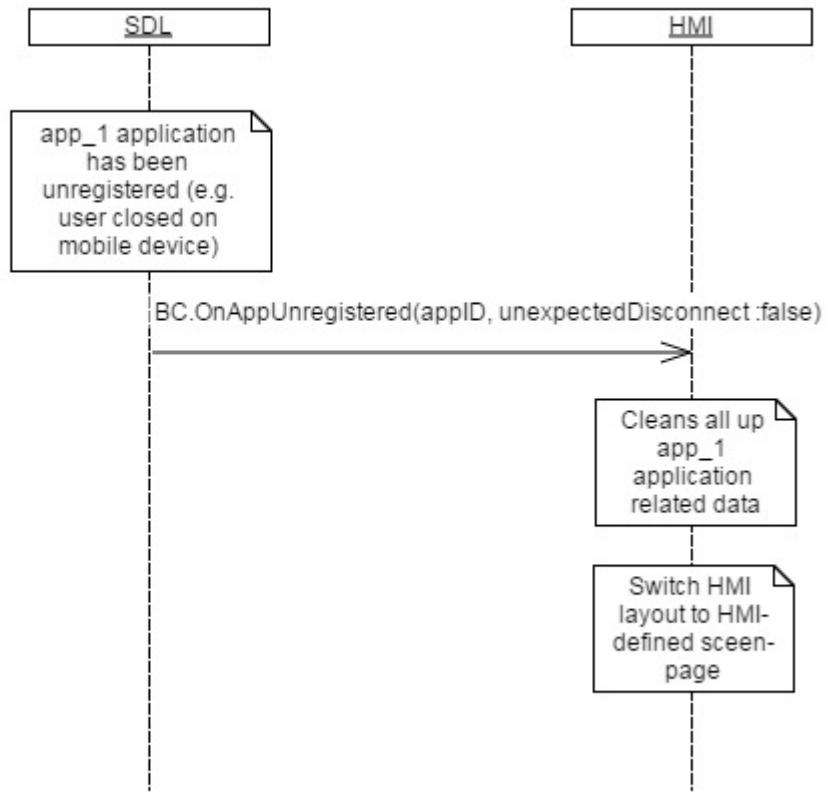
The information about the application (name, ID, etc.) is provided by SDL via either [UpdateAppList](#) or [OnAppRegistered](#) RPCs.

6.14.1.1 Parameters

Param Name	Type	Mandatory	Description
appID	Integer	true	ID of the application having been unregistered.
unexpectedDisconnect	Boolean	true	Defines whether the connection with the app was unexpectedly lost or this was the regular app's exit.

6.14.2 Sequence Diagrams

6.14.2.1 OnAppUnregistered of active application



6.14.3 JSON Messages Examples

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

6.15 OnExitApplication

6.15.1 Description

Type:	Notification
Sender:	HMI
Purpose:	Inform SDL that an application must be unregistered/put to NONE because of some specified reason on HMI

HMI must:

- 1) Provide the possibility for the User to EXIT any of registered applications (e.g. by HMI-defined button, VR command, etc.).

- 2) Track that the application being active or in background on HMI complies with the DRIVER DISTRACTION rules of the system (specific to each system)
 - 3) Track if the transport type corresponds to the application running (Navigation applications aren't allowed to be run on Bluetooth transport)
 - 4) Ignore all SDL API related to the the application defined as prohibited to be run on specified transport type
 - 5) Send `OnExitApplication` notification to SDL when
 - a) The User chooses to exit the application in one of HMI-suggested ways.
 - b) Some of DRIVER DISTRACTION rules applied on the system and require to switch the application into HMI NONE level
 - c) Navigation application is registered via BT transport type not allowed to be used for this application type
`(UNAUTHORIZED_TRANSPORT_REGISTRATION reason)`
- 6) Switch layouts according to workflow after application deactivation

Note:

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

6.15.1.1 Parameters

Param Name	Type	Mandatory	Description
reason	Common. <code>ApplicationExitReason</code>	true	The reason why the named application should be unregistered/put to NONE HMI Level
appID	Integer	true	ID of the application chosen to be unregistered/put to NONE.

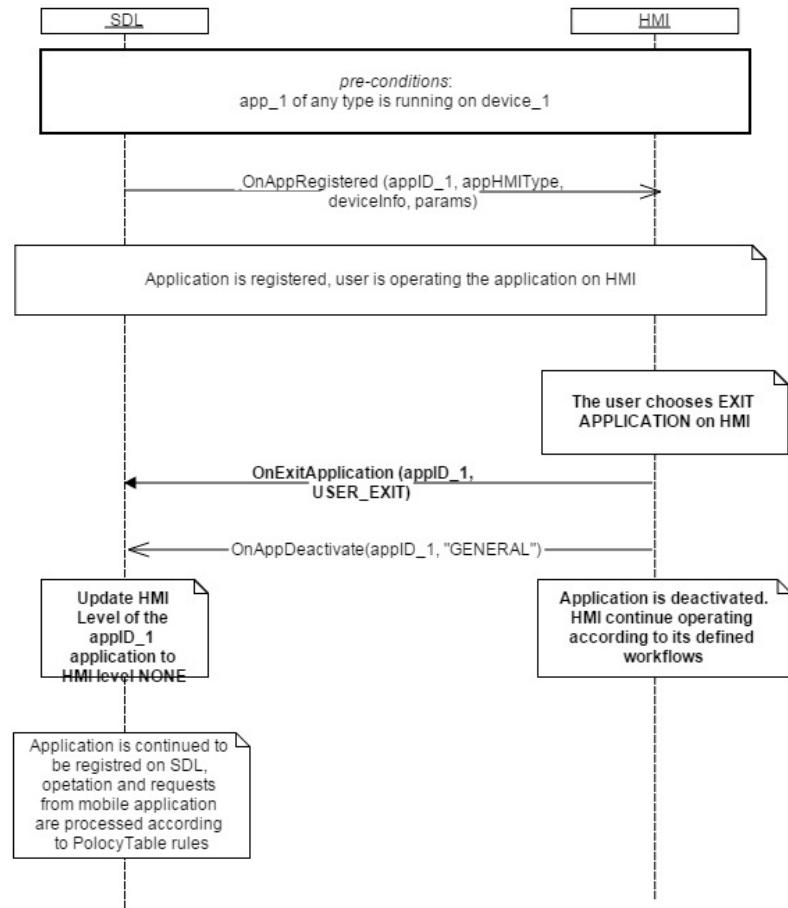
6.15.1.2 ApplicationExitReason

Element name	Value	Short Description
DRIVER_DISTRACTION_VIOLATION	0	Driver distraction rules violating observed. On getting this value, SDL puts the named app to NONE HMILevel</description>
USER_EXIT	1	User made Exit choice for the named application on HMI. On getting

Element name	Value	Short Description
		this value, SDL switches the named application to NONE HMILevel
UNAUTHORIZED_TRANSPORT_REGISTRATION	2	On getting this value, SDL unregisters the named application because transport type is not compatible with the application which uses it

6.15.2 Sequence Diagrams

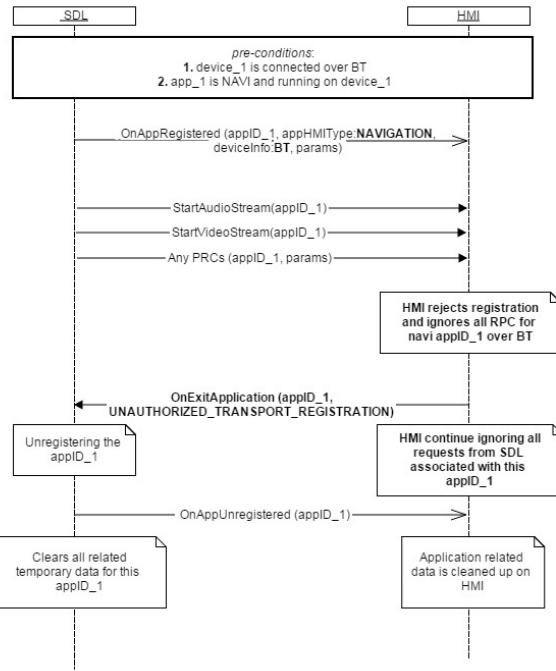
6.15.2.1 OnExitApplication USER_EXIT



6.15.2.2 OnExitApplication

UNAUTHORIZED_TRANSPORT_REGISTRATION

N



6.15.3 JSON Messages Examples

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

6.16 OnExitAllApplications

6.16.1 Description

Type:	Notification
Sender:	HMI
Purpose:	Inform about the event that causes exit of all applications.

SDL requires this notification to accurately close the sessions with the registered applications before reloading/shutting down by the reason of User's actions.

HMI must:

- Send the notification with appropriate "reason" upon one of the following results of User's actions:

- ACC key position set-up (for the both SUSPEND and IGNITION_OFF cases, see the diagrams [6.16.2.1 OnExitAllApplications SUSPEND and IGNITION OFF](#))
- Master reset

Return to factory default settings (HU system is expected to be rebooted).

Note: SDL ignores all invalid notifications which come from HMI (invalid JSON, invalid data types/bounds etc)

6.16.1.1 Parameters

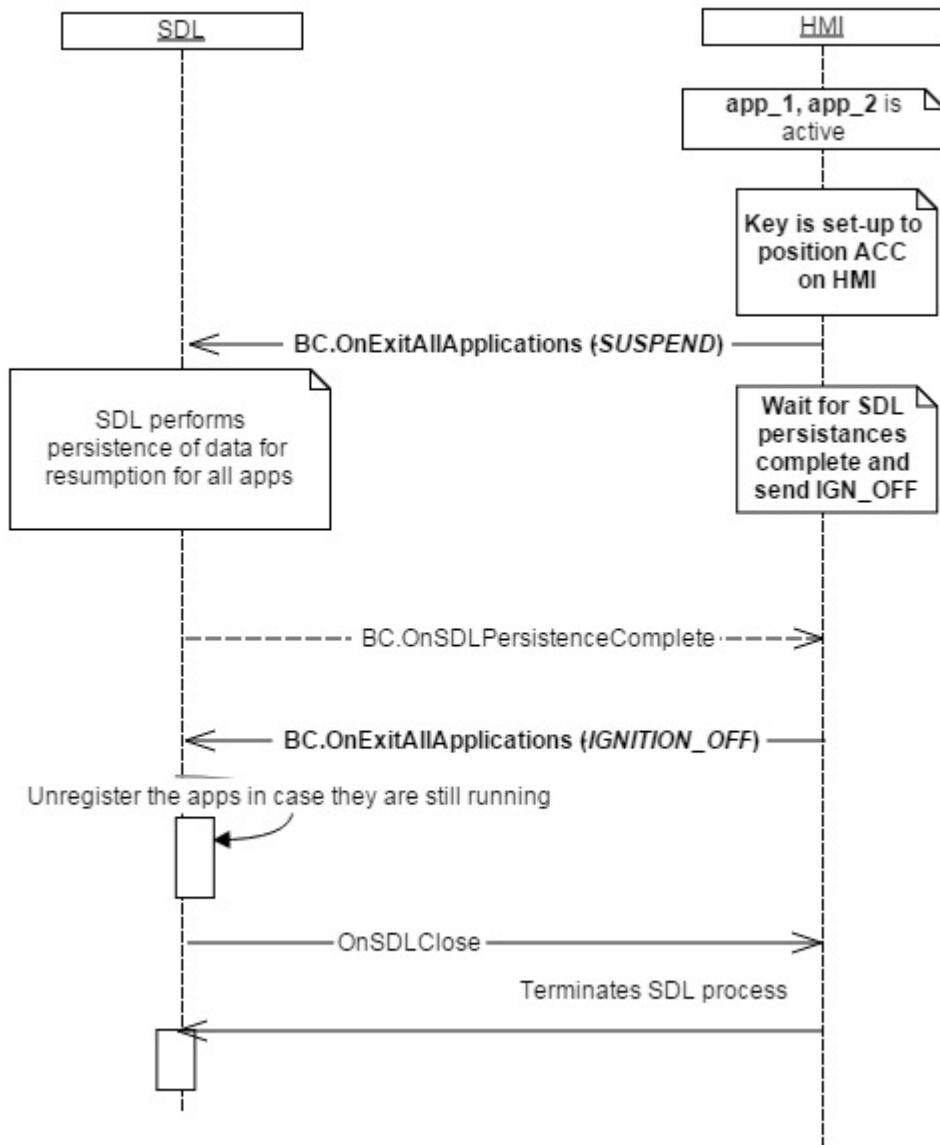
Param Name	Type	Mandatory	Description
reason	Common.Application.sCloseReason	true	The reason for exiting all the applications. See ApplicationsCloseReason.

6.16.1.2 ApplicationsCloseReason Enumeration

Element name	Short Description
IGNITION_OFF	Sent after notification with SUSPEND reason. When SDL persisted it's data, OnExitAllApplications(IGNITION_OFF) must be sent by HMI to notify SDL about SDL closure process. See 6.16.2 Sequence Diagrams for more details
MASTER_RESET	Master reset of Head Unit system
FACTORY_DEFAULTS	Return to factory defaults.
SUSPEND	Key position is ACC. Notifies SDL to perform persistence of data for resumption.

6.16.2 Sequence Diagrams

6.16.2.1 OnExitAllApplications SUSPEND and IGNITION OFF



6.16.3 JSON Messages Examples

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

6.17 PlayTone

6.18.1 Description

Type:	Notification
Sender:	SDL
Purpose:	Play HMI standard notifying sound.

This notification may follow some SDL request that brings changes on HMI and is desirable to be noticed or reacted by the User (e.g. alert message displayed, audio capturing started etc.).

6.17.2 Request

6.17.2.1 Parameters

HMI must:

- Play HMI standard notifying sound for appropriate type of the request which is defined in “methodName” parameter

Param Name	Type	Mandatory	Description
appID	Integer	true	ID of the application that invoked this notification
methodName	Common.MethodName	true	Defines the type of the request which initiates playing a tone

6.17.2.2 MethodName

Element name	Value	Short Description
ALERT	0	Defines that “Alert” mobile request initiates a current PlayTone
SPEAK	1	Defines that “Speak” mobile request initiates a current PlayTone
AUDIO_PASS_THROUGH	2	Defines that “PerformAudioPassThru” mobile request which initiates a current PlayTone
ALERT_MANEUVER	3	Defines the type of the mobile API request (AlertMeneuver mobile API) which initiates the request on HMI e.g. (TTS.Speak, BC.PlayTone)

6.17.3 Response

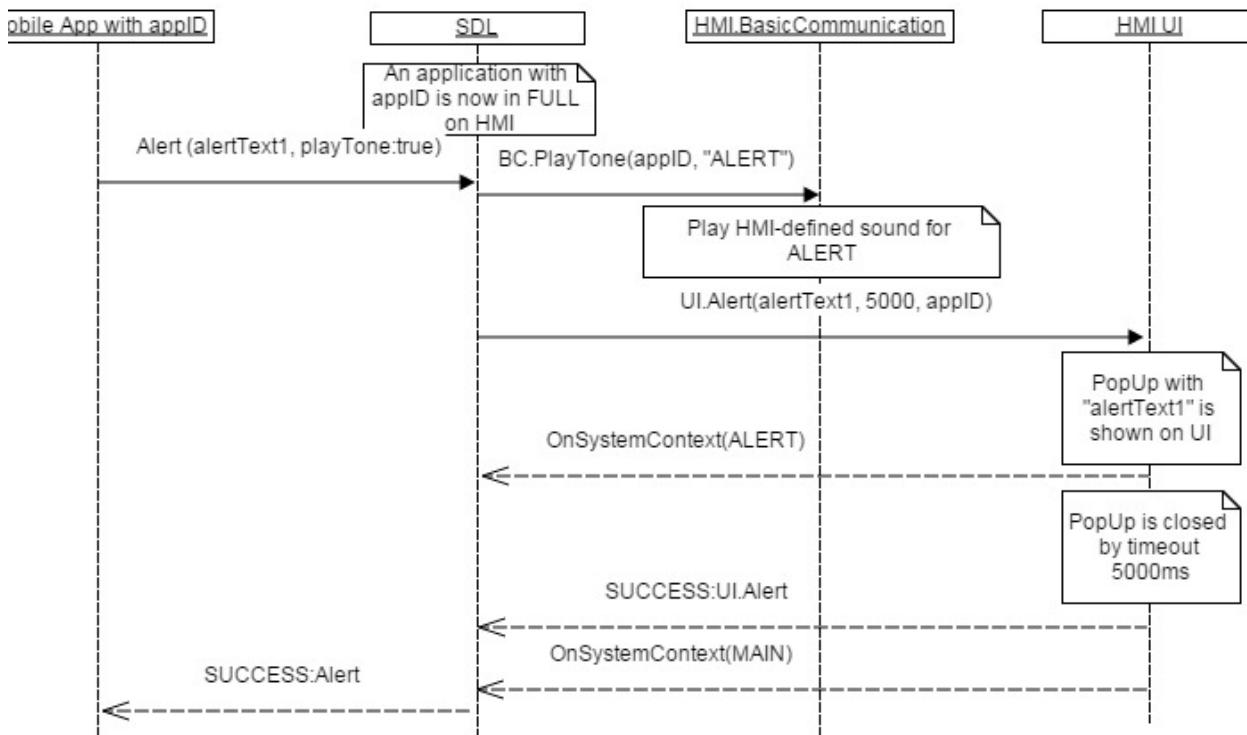
Note:

There is a difference in message type for WebSocket and D-Bus connection (see the table below).

Result	Description	Message type		Message Params	Notes
		WebSocket	D-Bus		
Success	SUCCESS	JSON response	Regular response	allow, Code: 0	See section 6.6.3.2 The information is available and provided.
Failure	INVALID_DATA	JSON error message	Regular response	Code: 11	appID is out of range
	INVALID_ID			Code: 13	The app with the appID doesn't exist on HMI
	GENERIC_ERROR			Code: 22	Some failure occurred.

6.17.4 Sequence Diagrams

6.17.4.1 PlayTone with Alert request



6.17.5 JSON Messages Examples

6.17.5.1 Request

```
{
    "jsonrpc" : "2.0",
    "method" :
"BasicCommunication.PlayTone"
"params" :
{
    "appId" : 123,
    "methodName": "ALERT"
}
}
```

6.17.5.2 Response

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

```
    }
}
```

6.17.5.3 Error message

```
{
  "id" : 47,
  "jsonrpc" : "2.0",
  "error" :
  {
    "code" : 22,
    "message" : "Unknown system error",
    "method" :
"BasicCommunication.PlayTone"
  }
}
```

6.18 OnSystemRequest

6.18.1 Description

Type:	Notification
Sender:	HMI
Purpose:	A request to mobile app for initiating the download of data.

There're 2 types of using OnSystemRequest:

- As an asynchronous request from the system for a specific data from the device or the cloud
- As a response to a request from the device or cloud. Binary data can be included in a hybrid part of message for some requests (such as Authentication request responses)
- As a request to mobile side to stop current uploading and data transferring via PutFile to HMI

RequestType defines the type of the requested data from a mobile device or a cloud. The module may request this data but it's SDL's responsibility to block SystemRequest in case it intends to transfer the data of the type not allowed by Policy Table.

To inform HMI about requestTypes allowed the following HMI_API commands are also used:

- [OnAppPermissionChanged](#) – sent from SDL to HMI when a PT update comes in and changes the app policy.
- [OnAppRegistered](#) – sent from SDL to HMI when an app registers with SDL

Note In case HMI sends OnSystemRequest with a RequestType disallowed by PolicyTable, SDL will ignore it.

Also SDL ignores all invalid notifications which come from HMI (invalid JSON, invalid data types/bounds etc)

HMI must:

- 1) Send OnSystemRequest in case some specific data is requested from mobile device/cloud or some binary data need to be sent to mobile side

6.18.1.1 Parameters

Param Name	Type	Mandatory	Additional	Description
requestType	Common.RequestType	true	-	The type of system request.
url	string	false	minlength = 1 maxlength = 1000	Optional array of URL(s) for HTTP requests.
fileType	Common.FileType	false	-	Optional file type (meant for HTTP file requests).
offset	Integer	false	minvalue = 0 maxvalue = 100000000000	Optional offset in bytes for resuming partial data chunks
length	Integer	false	minvalue = 0 maxvalue = 100000000000	Optional length in bytes for resuming partial data chunks.
timeout	Integer	false	minvalue = 0 maxvalue = 2000000000	
fileName	string	true	minlength = 1 maxlength = 255	File reference name.
appID	Integer	false		Internal ID of the application that corresponds to the policyAppID

6.18.1.2 RequestType Enumeration

Element name	Value	Short 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	

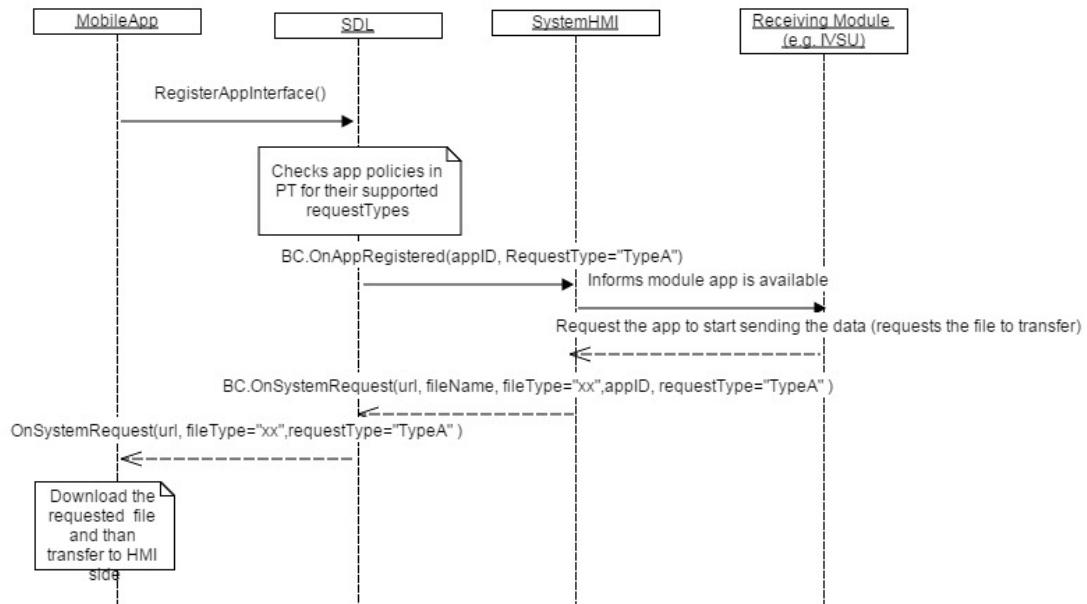
Element name	Value	Short Description
MEDIA	18	
FOTA	19	

6.18.1.3 FileType Enumeration

Element name	Value	Short Description
GRAPHIC_BMP	0	
GRAPHIC_JPEG	1	
GRAPHIC_PNG	2	
AUDIO_WAVE	3	
AUDIO_MP3	4	
AUDIO_AAC	5	
BINARY	6	
JSON	7	

6.18.2 Sequence Diagrams

6.18.2.1 OnSystemRequest requests the file to download from Cloud



6.18.3 JSON Messages Examples

```
{
    "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":http://policies.telematics.ford.com/api/policies
  }
}

```

6.18.4 D-Bus Messages Examples



6.19 OnPutFile

6.19.1 Description

Type:	Notification
Sender:	SDL
Purpose:	Inform about application-related file has been uploaded into shared folder

OnPutFile notifies HMI that some file has been put to SharedFolder/or SystemFolder (for a PutFile with systemFile="true" parameter) and may be used by HMI.

HMI must:

- 1) Use the appropriate uploaded file according to it's workflow (IVSU, SystemRequest, affected RPCs)
- 2) Show the icon in case the RPC's data which requested this icon is already displayed on the screen and OnPutFile is obtained after this RPC came to HMI
- 3) Show the icon which came before the correspondant RPC right after the RPC which uses mentioned icon will come and it's data will be displayed

For more details please refer [6.19.2 Sequence Diagrams OnPutFile](#).

The list of the RPCs and the data structures on which OnPutFile affects on:

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

SDL Note: In case an application sends images after the request timeout of the following RPCs, SDL must not move the icons into SharedFolder and just store them the application folder. The exceptional RPCs and parameters are:

- 1) Alert (SoftButton)
- 2) ScrollableMessage (SoftButton)
- 3) UpdateTurnList (SoftButton)

6.19.1.1 Parameters

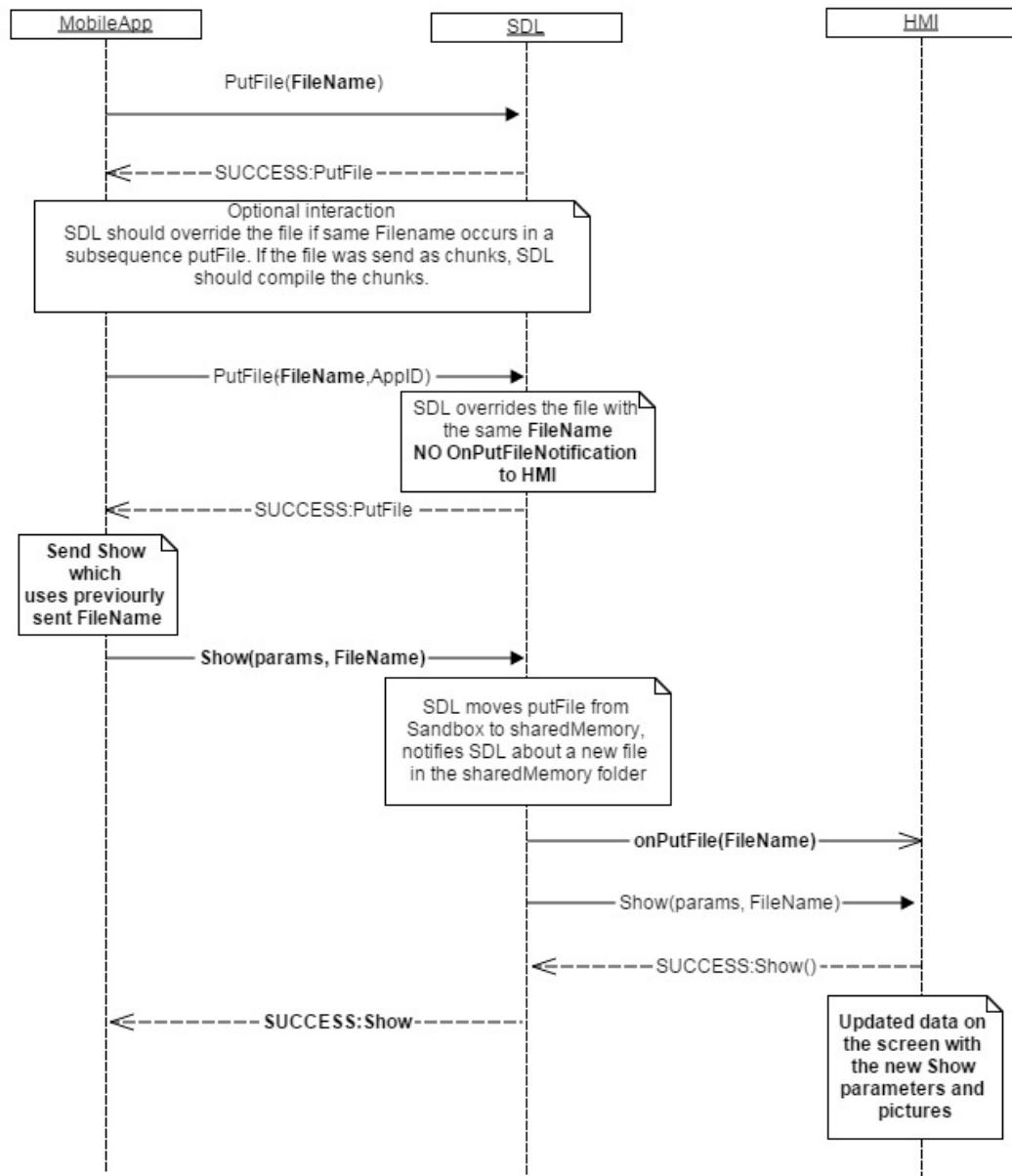
Param Name	Type	Mandatory	Additional	Description
offset	Integer	false	minvalue = 0 maxvalue = 100000000000	Optional offset in bytes for resuming partial data chunks
length	Integer	false	minvalue = 0 maxvalue = 100000000000	Optional length in bytes for resuming partial data chunks
fileSize	Integer	false	minvalue = 0 maxvalue = 100000000000	Full Size of file. sends in first OnPutFile notification if file is splitted into many PutFiles
FileName	String	true	maxlength = 255	File references filename to a systemFile (obtained via PutFile with systemFile=true).
syncFileName	String	true	maxlength = 255	File reference name.
fileType	Common.FileType	true	-	Selected file type.
persistentFile	Boolean	false	-	<p>Indicates if the file is meant to persist between sessions / ignition cycles. If set to TRUE, then the system will aim to persist this file through session / cycles. While files with this designation will have priority over others, they are subject to deletion by the system at any time. In the event of automatic deletion by the system, the app will receive a rejection and have to resend the file. If omitted, the value will be set to false.</p>

6.19.1.2 FileType Enumeration

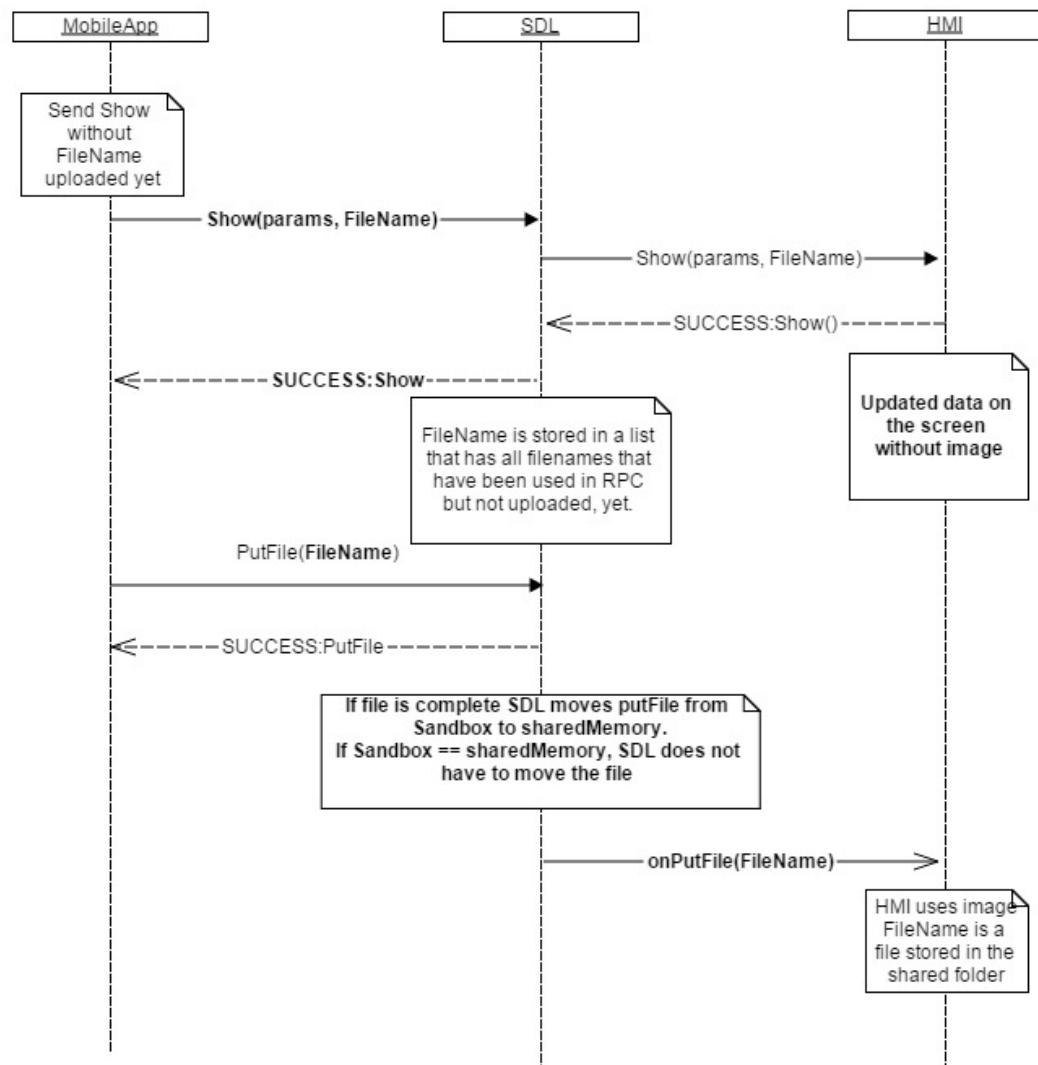
Element name	Value	Short Description
GRAPHIC_BMP	0	
GRAPHIC_JPEG	1	
GRAPHIC_PNG	2	
AUDIO_WAVE	3	
AUDIO_MP3	4	
AUDIO_AAC	5	
BINARY	6	
JSON	7	

6.19.2 Sequence Diagrams

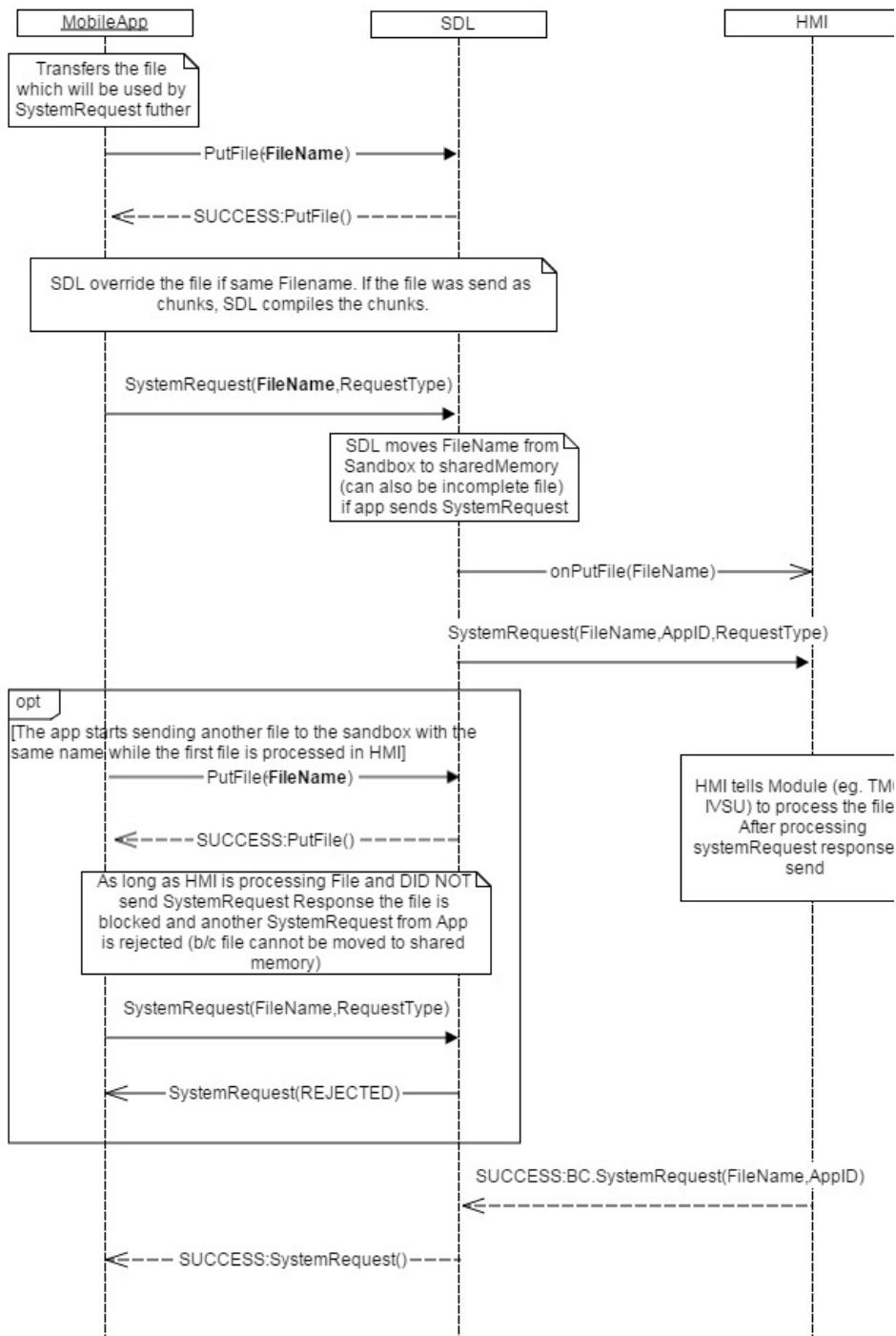
6.19.2.1 OnPutFile. File is uploaded before RPC is used



6.19.2.2 OnPutFile. File is uploaded after RPC is used



6.19.2.3 OnPutFile. File uploading via SystemRequest



6.19.3 JSON Messages Examples

```
{
  "jsonrpc" : "2.0",
  "method" : "file.upload",
  "params" : {
    "filename": "testfile.txt"
  },
  "id" : 1
}
```

```

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

```

6.20 OnFileRemoved

6.20.1 Description

Type:	Notification
Sender:	SDL
Purpose:	Inform about application-related file removing

Each application file which is used by applications must be sent to SDL via PutFile by the application. The file is stored in the appropriate folder accessible by SDL only. In case the file must be used on HMI, SDL transfers the file into appropriate application shared folder.

When the file is requested to be removed by the application, SDL must notify HMI that it has already cleaned application data and removed appropriate file from the application SharedFolder. (e.g. then HMI should change the icon to the default one and clean up its icon references)

SDL note: SDL notifies HMI about files removal only if they were moved into HMI shared folder before (or SDL application folder equals HMI SharedFolder).

HMI must:

- 1) Change the icon to the default one or not to display any picture instead of removed file (depends on HMI)

6.20.1.1 Parameters

Param Name	Type	Mandatory	Additional	Description
fileName	String	true	minlength = 1 maxlength = 30	The full path to the file that has been removed
fileType	Common.FileType	true	-	The file type. See FileType.
appID	Integer	true	-	ID of the application.

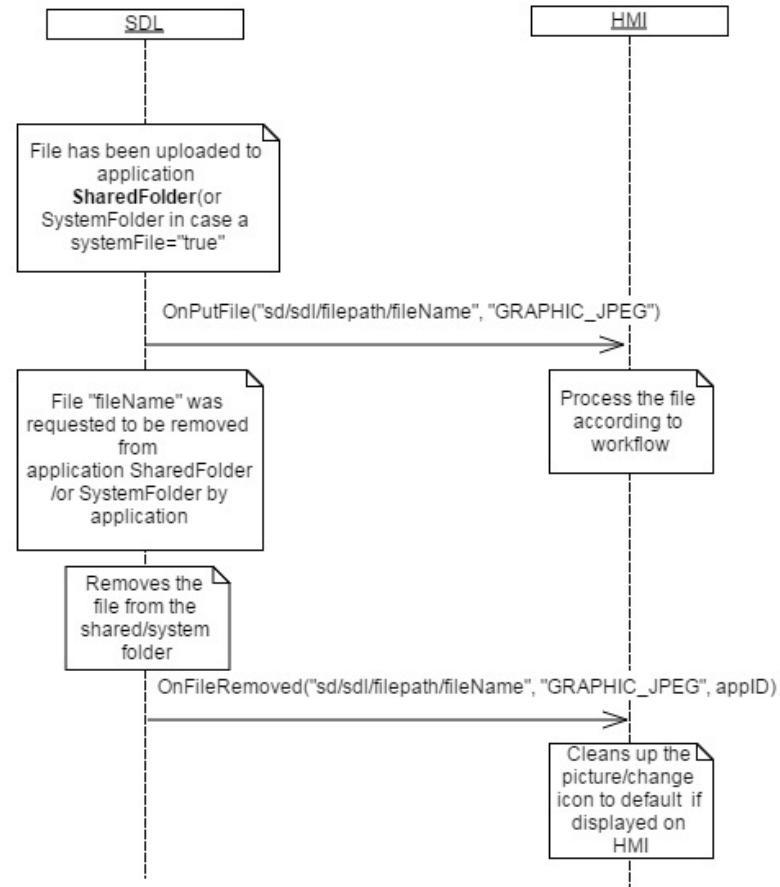
6.20.1.2 FileType Enumeration

Element name	Value	Short Description
GRAPHIC_BMP	0	
GRAPHIC_JPEG	1	
GRAPHIC_PNG	2	

Element name	Value	Short Description
AUDIO_WAVE	3	
AUDIO_MP3	4	
AUDIO_AAC	5	
BINARY	6	
JSON	7	

6.20.2 Sequence Diagrams

6.20.2.1 OnFileRemoved



6.20.3 JSON Messages Examples

```
{
    "jsonrpc" : "2.0",
    "method" :
"BasicCommunication.OnFileRemoved"
    "params" :
{
    "appID" : 31780,
    "fileName" :
"/fs/rwdata/storage/sdl/Emergency5844
21907/syncFileName",
  
```

```
    "fileType" : "GRAPHIC_BMP"
}
}
```

<FORDSPECIFIC>

6.21 OnSDLClose

6.21.1 Description

Type:	Notification
Sender:	SDL
Purpose:	Inform about SDL has ended its activities and ready to be terminated by System

OnSDLClose notification received from SDL means that SDL has terminated its activities and notifies HMI that all necessary data is saved/cleaned up, applications are unregistered and the process is ready to be closed by system.

HMI must:

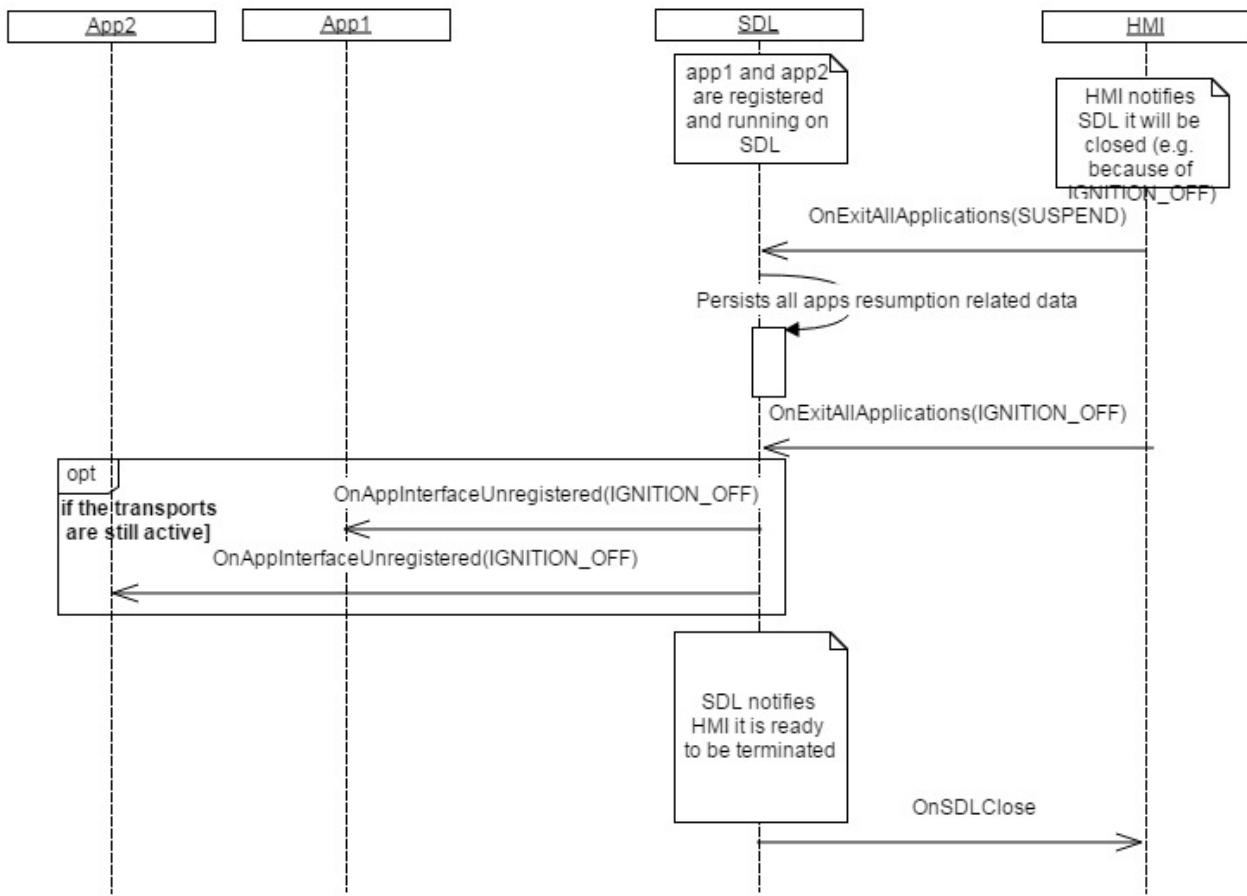
- 1) Notify SDL about intention to terminate it because of one of the reasons (IGNITION_OFF /FACTORY_RESET /MASTER_RESET) via [OnExitAllApplications](#)
- 2) Terminate SDL process only after OnSDLClose is received. Otherwise, application data may be lost on SDL side

SDL information:

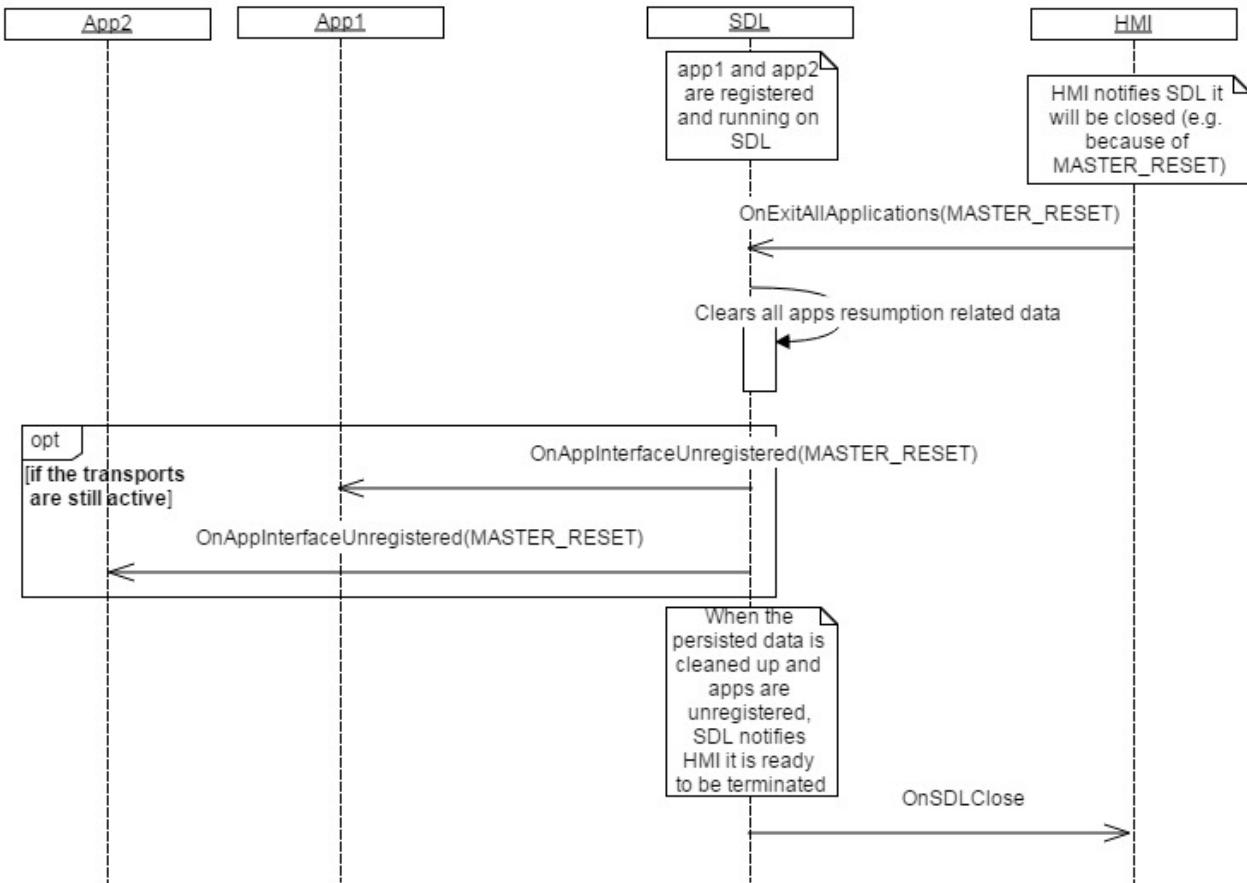
- 1) SDL unregisters the apps if the transports are still available in case of IGNITION_OFF reason
- 2) SDL cleans up applications persistant data in case of FACTORY_RESET/MASTER_RESET reason of OnExitAllApplications

6.21.2 Sequence Diagrams

6.21.2.1 OnSDLClose IGNITION_OFF



6.21.2.1 OnSDLClose MASTER_RESET



6.21.3 JSON Messages Examples

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

6.22 OnUpdateDeviceList

6.22.1 Description

Type:	Notification
Sender:	HMI
Purpose:	Get the updated list of found devices.

On receipt of this notification SDL provides the list of devices found during the recent search procedure . Updated device list is being sent to HMI via UpdateDeviceList RPC.

HMI must:

- 1) Provide the User with the possibility to initiate the device list updating in one of accessible ways (VR, buttons, etc.).

- 2) Send `OnUpdateDeviceList` notification when the User has chosen to update the device list.

Note:

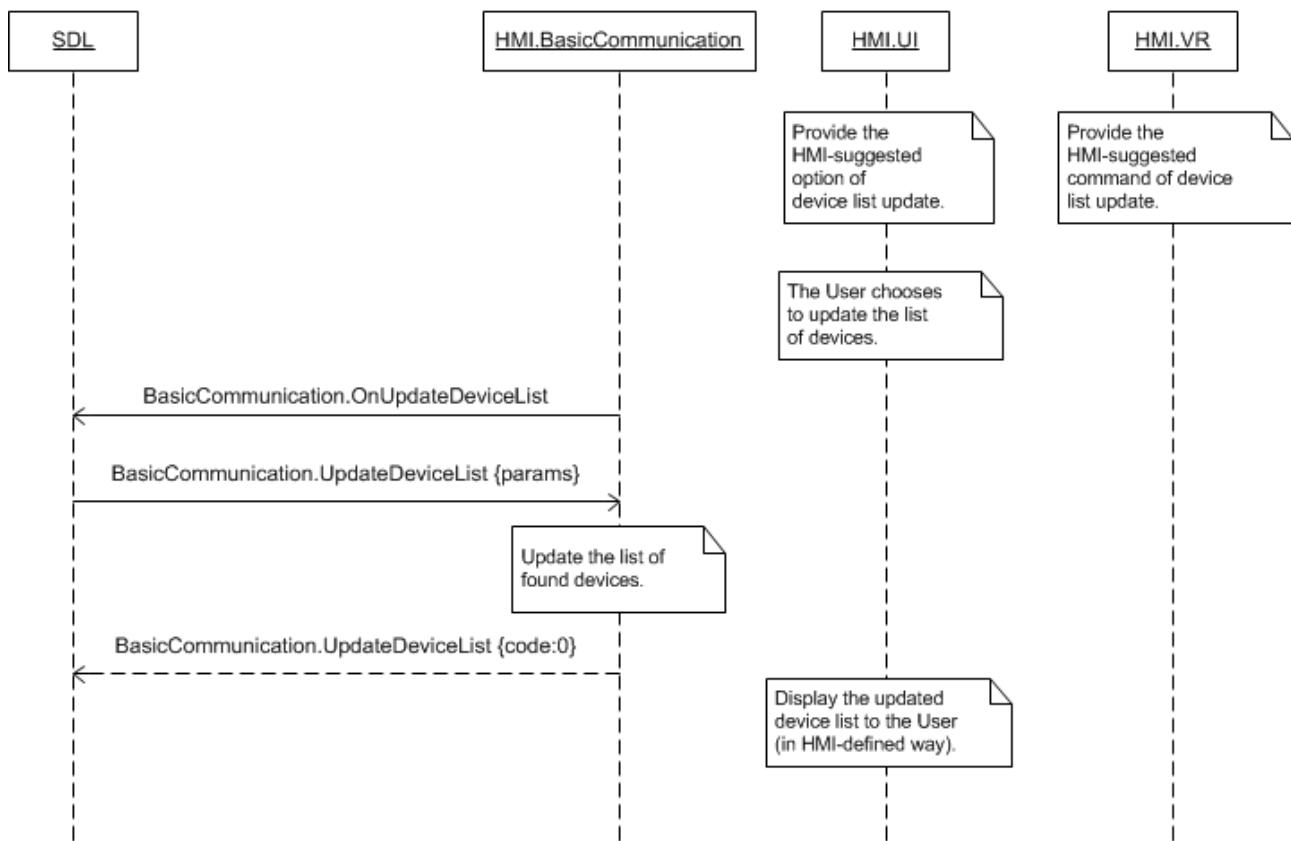
The difference:

- `OnUpdateDeviceList` results `SDL` to provide the updates of the recent search.
- `OnStartDeviceDiscovery` initiates `SDL` to start the new search procedure

Note: `SDL` ignores all invalid notifications which come from `HMI` (invalid JSON, invalid data types/bounds etc)

6.22.2 Sequence Diagrams

6.22.2.1 `OnUpdateDeviceList` upon User's request and resulting `UpdateDeviceList`



6.22.3 JSON Messages Examples

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

6.23 OnResume AudioSource

6.23.1 Description

Type:	Notification
Sender:	SDL
Purpose:	Inform about the named application being resumed needs to become audible on HMI

SDL sends OnResume AudioSource to HMI only in case when as a result of AudioSource 'resumption' procedure SDL has detected the application had the state of 'LIMITED, AUDIBLE' on HMI before the previous ignition off.

In case of successful HMIlevel resumption, SDL first assigns 'default_hmi' level from policies and after **3sec timeout**, resumes the application either to FULL or LIMITED (whichever applicable). **Important SDL note:** When SDL resumes the application into FULL HMIlevel with **AUDIBLE** AudioStreamingState, OnResume AudioSource **MUST NOT** be sent to HMI. Only the following sequence is applicable:

1. SDL -> HMI:
BasicCommunication.OnAppRegistered
2. SDL ->
HMI: BasicCommunication.ActivateApp

Important note:

SDL performs AudioSource resumption only if the applications which met the both conditions:

1. Running 30 minutes prior to OnExitAllApplication(SUSPEND) received from HMI
2. Reconnecting back not later than 30 seconds after BC.OnReady is received after new ignition cycle started
SDL will postpone the HMIlevel resumption **in case** occurred during the **active phone call** (see the diagrams [6.23.2.3 – 6.23.2.4](#)):
 - a) first SDL will assign the default_hmi from policies
 - b) after the phone call ends, SDL will resume the HMIlevel of this application

HMI must:

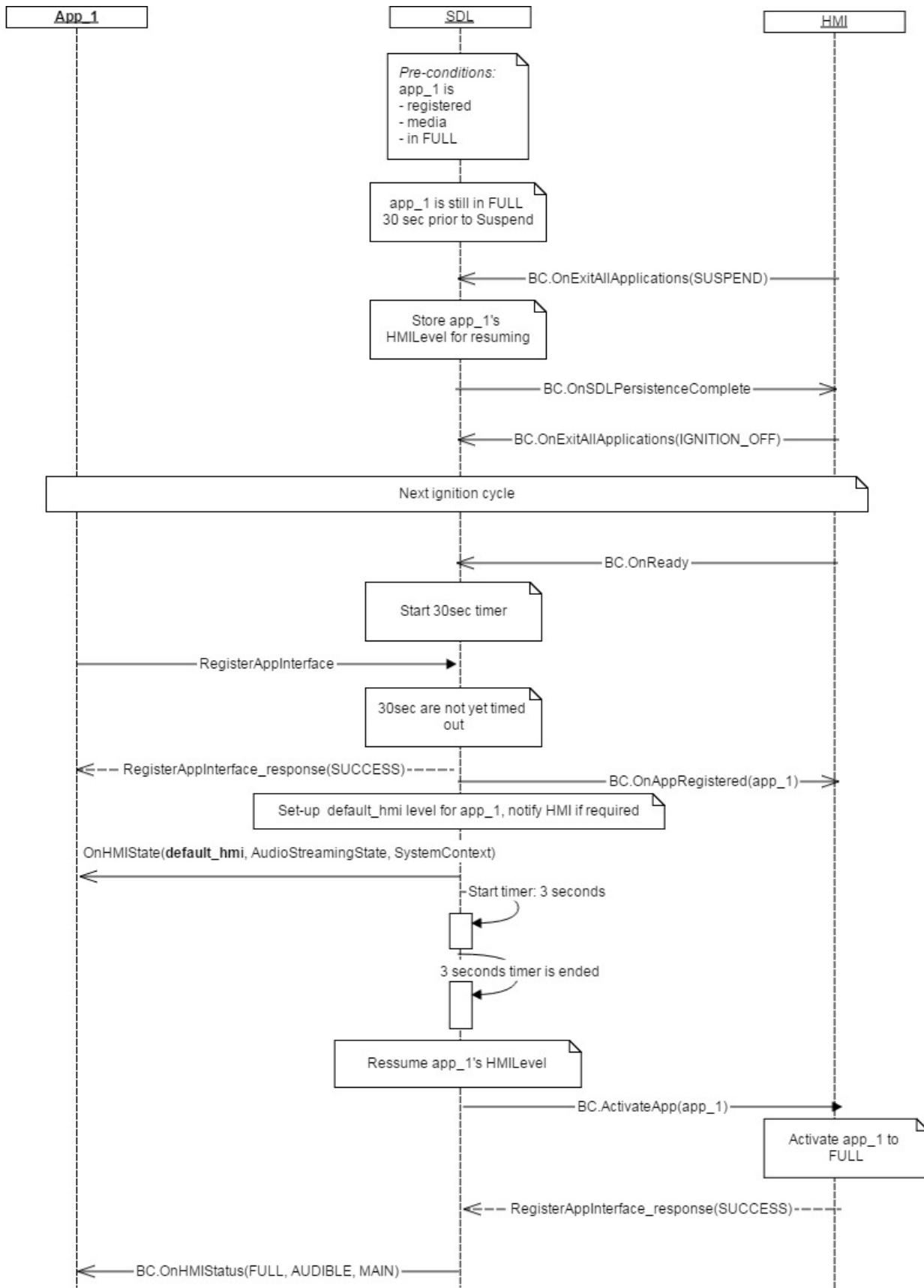
- 1) Activate only the audio source for the application with appID received.
- 2) Not activate the application itself (application's layout is not shown to the user). The named application must stay on screen layout's background on HMI.

6.23.1.1 Parameters

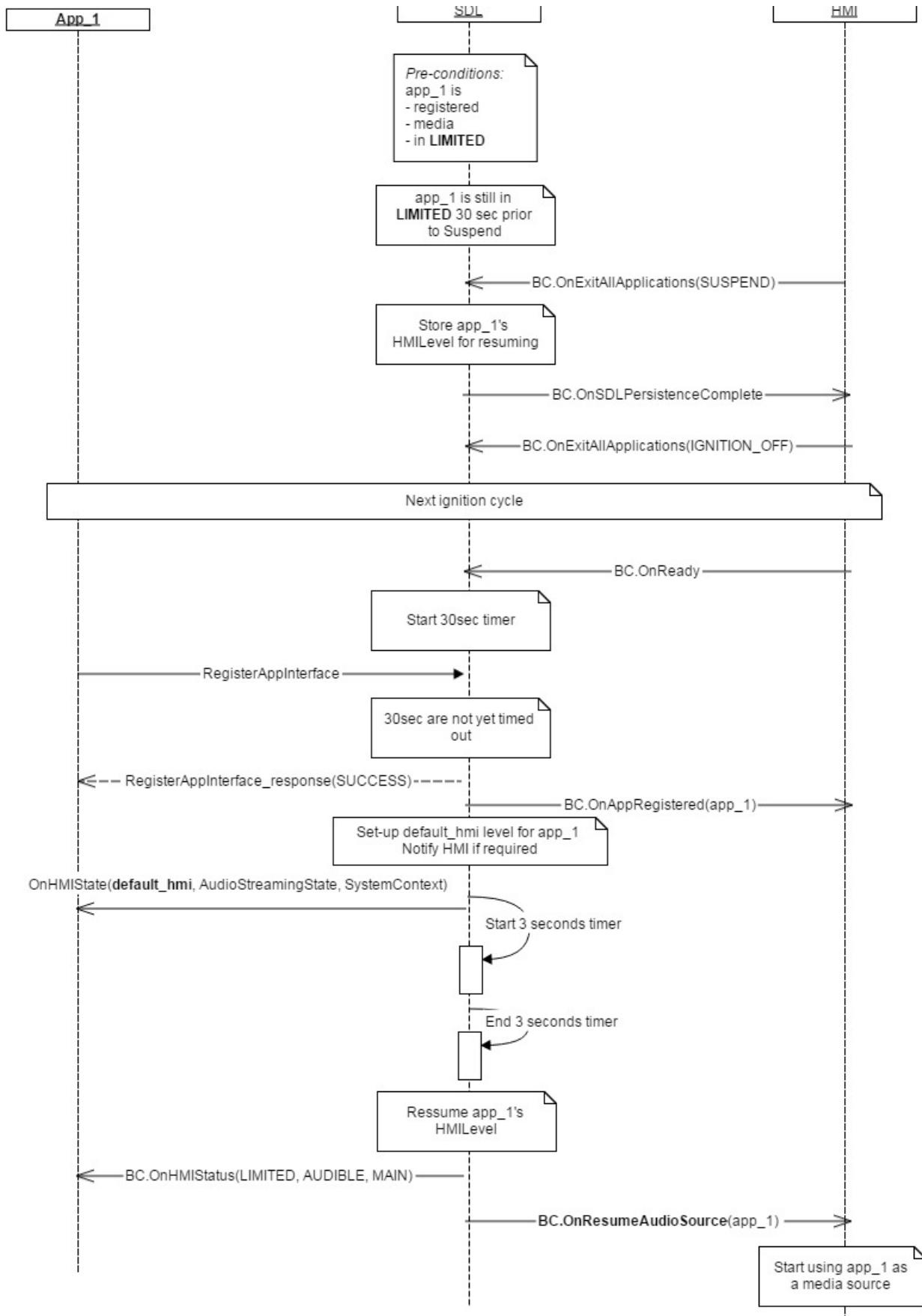
Param Name	Type	Mandatory	Additional	Description
appID	Integer	true	–	The ID of the application that HMI must activate the audio source for.

6.23.2 Sequence Diagrams

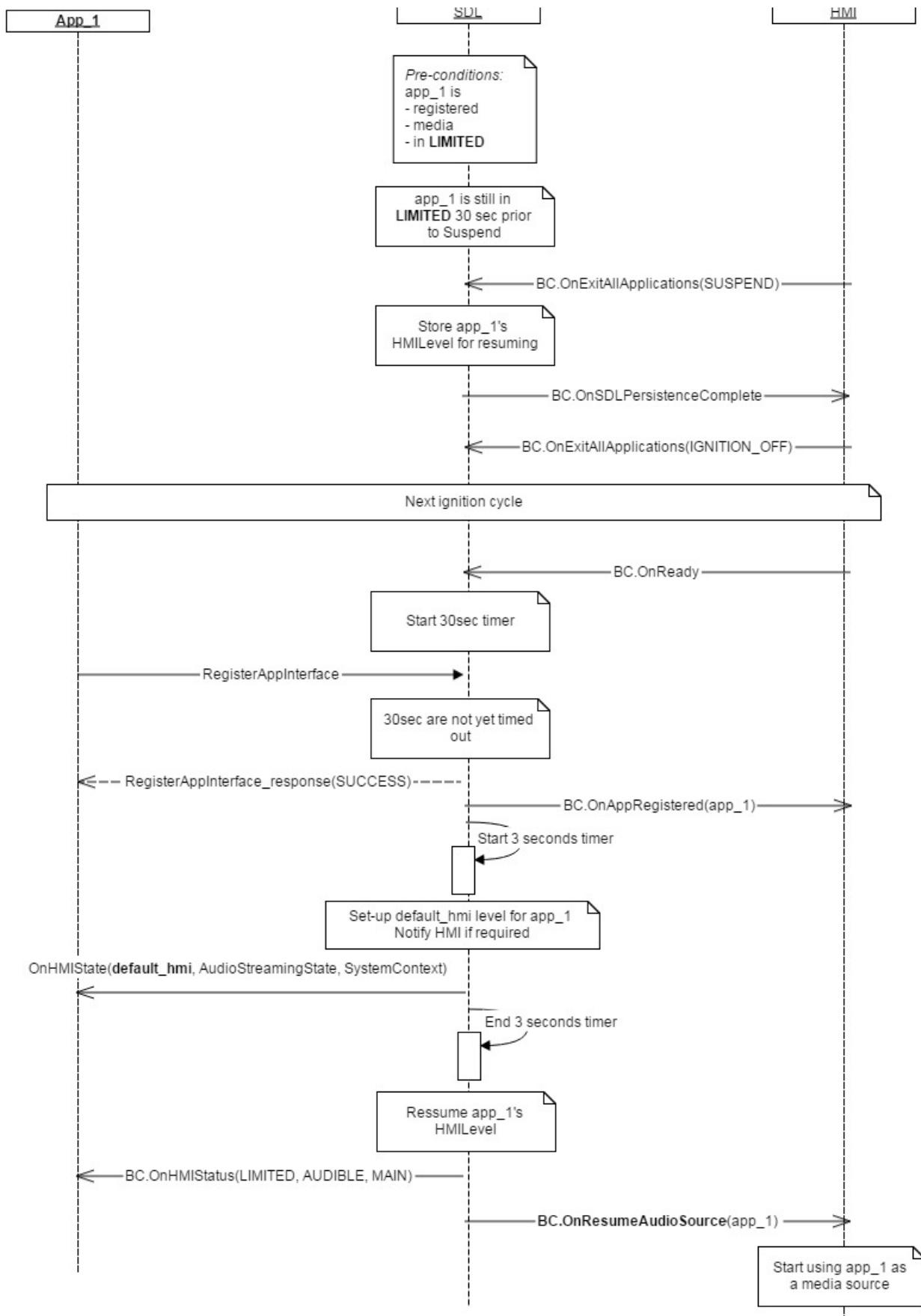
6.23.2.1 AudioSource resumption for the app with FULL HMI level



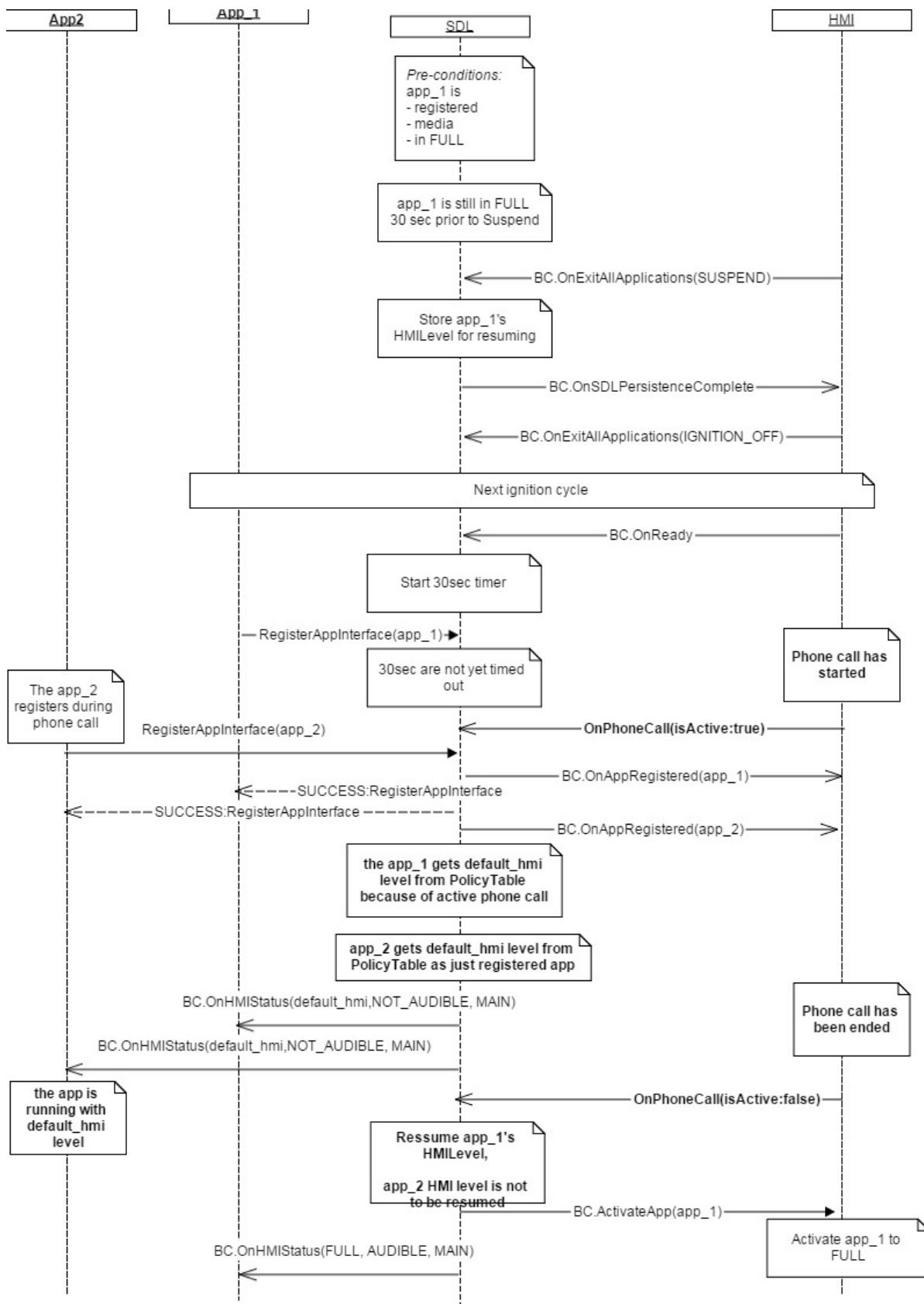
6.23.2.2 OnResume AudioSource for the app with LIMITED HMI Level



6.23.2.3 OnResume AudioSource for the app with LIMITED HMI Level (Phone call is active)



**6.23.2.4 AudioSource resume for the app_1
with FULL HMIlevel and app_2 just registered
while Phone call is active**



6.23.3 JSON Messages Examples

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

6.24 UpdateAppList

6.24.1 Description

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

The request may come:

- After SDL connected the device and the SDL-enabled application has registered itself with SDL successfully.
- Upon User's request sent from HMI via OnFindApplications notification.

6.24.2 Request

6.24.2.1 Behavior

HMI must:

- 1) Update the list of SDL registered applications.
- 2) Use the `appID` provided within this request when sending the definite notifications or requests (e.g. `ActivateApp`, `OnCommand` and other) related to the corresponding application to SDL.
- 3) Provide the User with the possibility to choose among the registered applications on UI.

Note:

SDL adds the VR synonyms of the registered application to HMI via `OnAppRegistered.notification` as far as the application has been registered

6.24.2.2 Parameters

Param Name	Type	Mandatory	Additional	Description
applications	Common.HMIApplcation	true	Array = true MinSize = 0 MaxSize = 100	The list of registered applications. The array contains the detailed information about each application: its name, ID, type and other. See <code>HMIApplcation</code> .

6.24.2.3 HMI Application Structure

Param Name	Type	Mandatory	Additional	Description
appName	String	true	Maxlength = 100	The mobile application name. It is unique over all applications.
ngnMediaScreenAppName	String	false	Maxlength = 100	Provides an abbreviated version of the application name (may be displayed on the NGN media screen). If not provided, the appName should be used instead (and may be truncated if too long)
icon	String	false	-	Path to the application icon stored on HU hard disc.
deviceName	String	true	-	The name of the device where the identified application is running on.
appID	Integer	true	-	The application ID that remains unique during the ignition cycle. This ID will be sent by SDL further and must be provided by HMI within all the RPCs related to this application.
hmiDisplayLanguageDesired	Common.Language	false	-	The language that the application intends to use. See Language.
isMediaApplication	Boolean	false	-	Indicates whether the application is a media or a non-media one. Only media applications are allowed by SDL to stream audio to HU that is audible outside of the BT media source.
appType	Common.AppHMIType	false	array = true Minsize = 1 maxsize = 100	The HMI may use this information for determining what functionality should be available for the application (e.g. NAVIGATION type of application will require displaying the information and not playing the audio). See AppHMIType.
requestType	Common.RequestType	false	Minsize = 0 Maxsize = 100 array="true"	The list of SystemRequest's RequestTypes allowed by policies for the named application (The app's SystemRequest sent with RequestType out of this list will get 'disallowed' response from SDL). If SDL sends an empty array - any RequestType is allowed for this app. If SDL omits this parameter - none RequestType is allowed for this app (either this is a pre-registered app or such is dictated by policies).

6.24.2.4 AppHMIType Enumeration

Element name	Short Description
DEFAULT	The application of default type.
COMMUNICATION	The application for communication
MEDIA	The media application
MESSAGING	The application of messaging type
NAVIGATION	The application of navigation type
INFORMATION	The application of information type
SOCIAL	The application of social type
BACKGROUND_PROCESS	The application does not require displaying the information
TESTING	The application of testing type
SYSTEM	The application of system type

6.24.3 Response

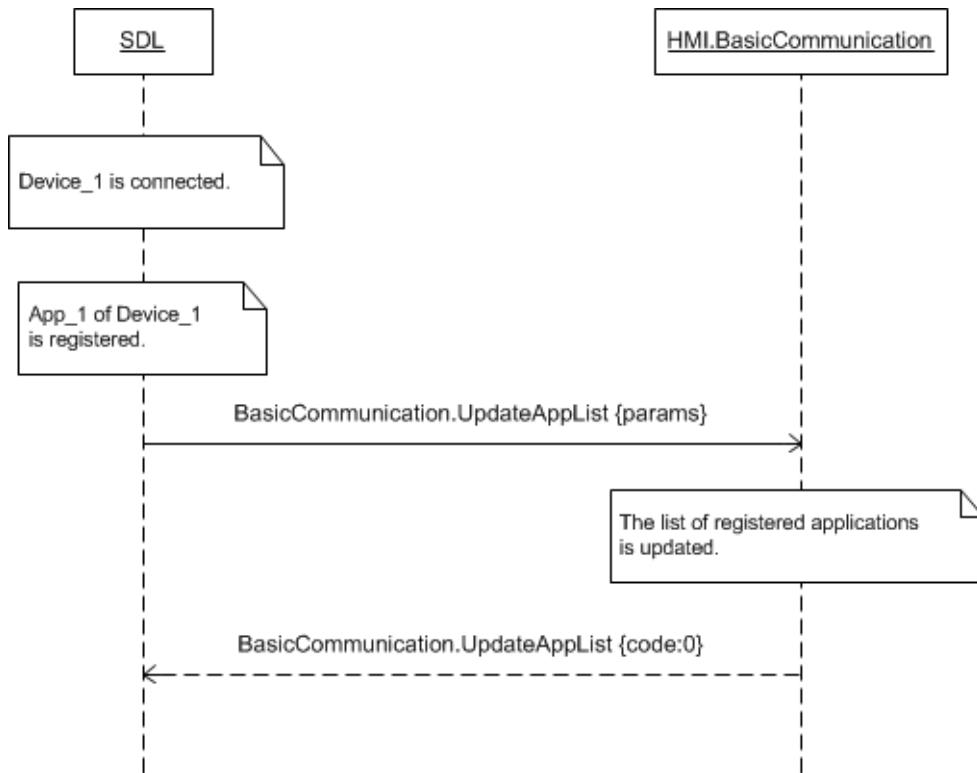
Note:

There is a difference in message type for WebSocket and D-Bus connection (see the table below).

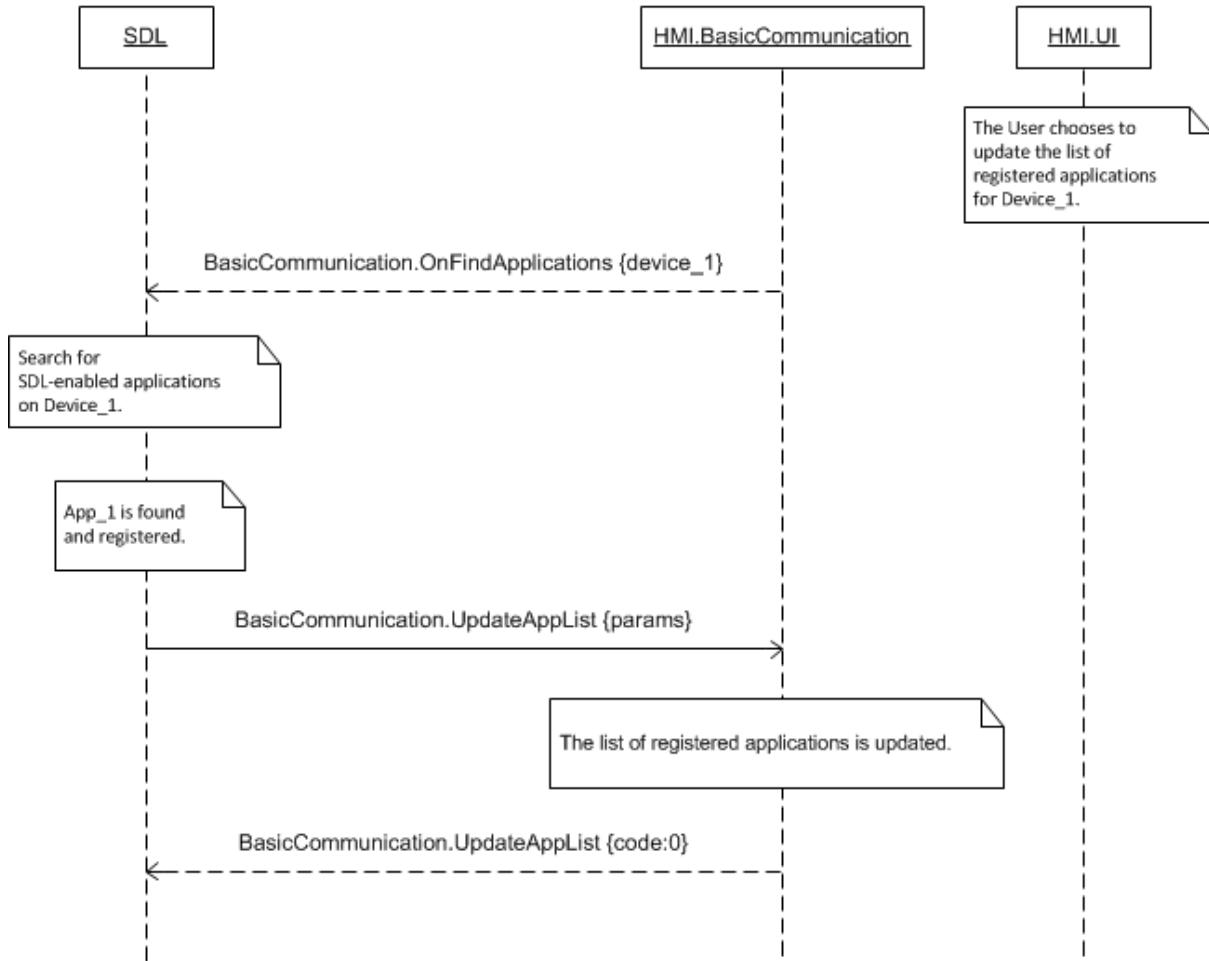
Result	Description	Message type		Message Params	Notes
		WebSocket	D-Bus		
Success	The HMI's list of registered applications is updated.	JSON response	Regular response	Code: 0	-
Failure	The HMI's list of registered applications is NOT updated	JSON error message	Regular response	Code: 1...23	Result code must correspond to the failure occurred.

6.24.4 Sequence Diagrams

6.24.4.1 UpdateAppList after application has just registered with SDL



6.24.4.2 UpdateAppList on User's Request (BT transport)



6.24.5 JSON Messages Examples

6.24.5.1 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
        }
    ]
}
}

```

6.24.5.2 Response

```

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

```

6.24.5.3 Error message

```

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

```

6.25 OnSDLPersistenceComplete

6.25.1 Description

Type:	Notification
Sender:	SDL
Purpose:	Inform about the applications data persistence are completed

SDL sends OnSDLPersistenceComplete right after all the data persistence operations are completed on SDL. This data is persisted for future resumption scenarios (e.g. in case SDL will get IGNITION_OFF from HMI). See [6.25.2.1 OnSDLPersistenceComplete diagram](#) for more information.

SDL Information:

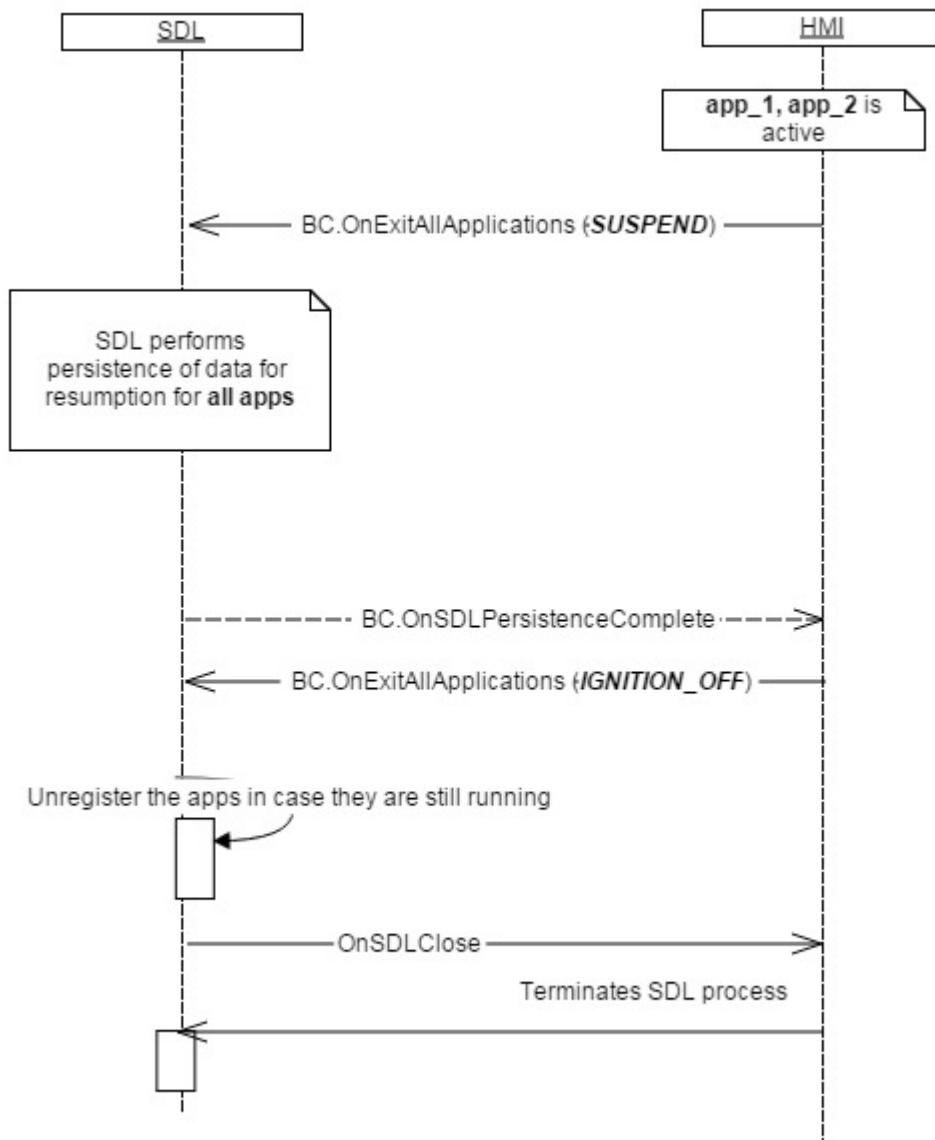
- When SDL gets `OnExitAllApplications{SUSPEND}` from HMI it persists the data for the case if IGNITION will be OFF

HMI must:

- 1) Send `OnExitAllApplications{SUSPEND}` to initiate data persistence process for the apps being registered on SDL
- 2) Wait for `OnSDLPersistenceComplete` before sending `OnExitAllApplications{IGNITION_OFF}`

6.25.2 Sequence Diagrams

6.25.2.1 OnSDLPersistenceComplete



6.25.3 JSON Messages Examples

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

6.26 OnPhoneCall

6.26.1 Description

Type:	Notification
Sender:	HMI->SDL
Purpose:	Notify SDL about Phone Call event started or ended

Information: When SDL receives BC.OnPhoneCall (isActive: true) from HMI, it changes the HMILevel of all applications currently in FULL and-or LIMITED to BACKGROUND. After getting OnPhoneCall (isActive: false) from HMI, SDL internally returns applications to the previous-to-phonecall HMILevel. SDL does not send BC.ActivateApp or BC.OnResumeAudioSource to HMI after the phone call is ended.

Note: SDL ignores all invalid notifications which come from HMI (invalid JSON, invalid data types/bounds etc)

HMI must:

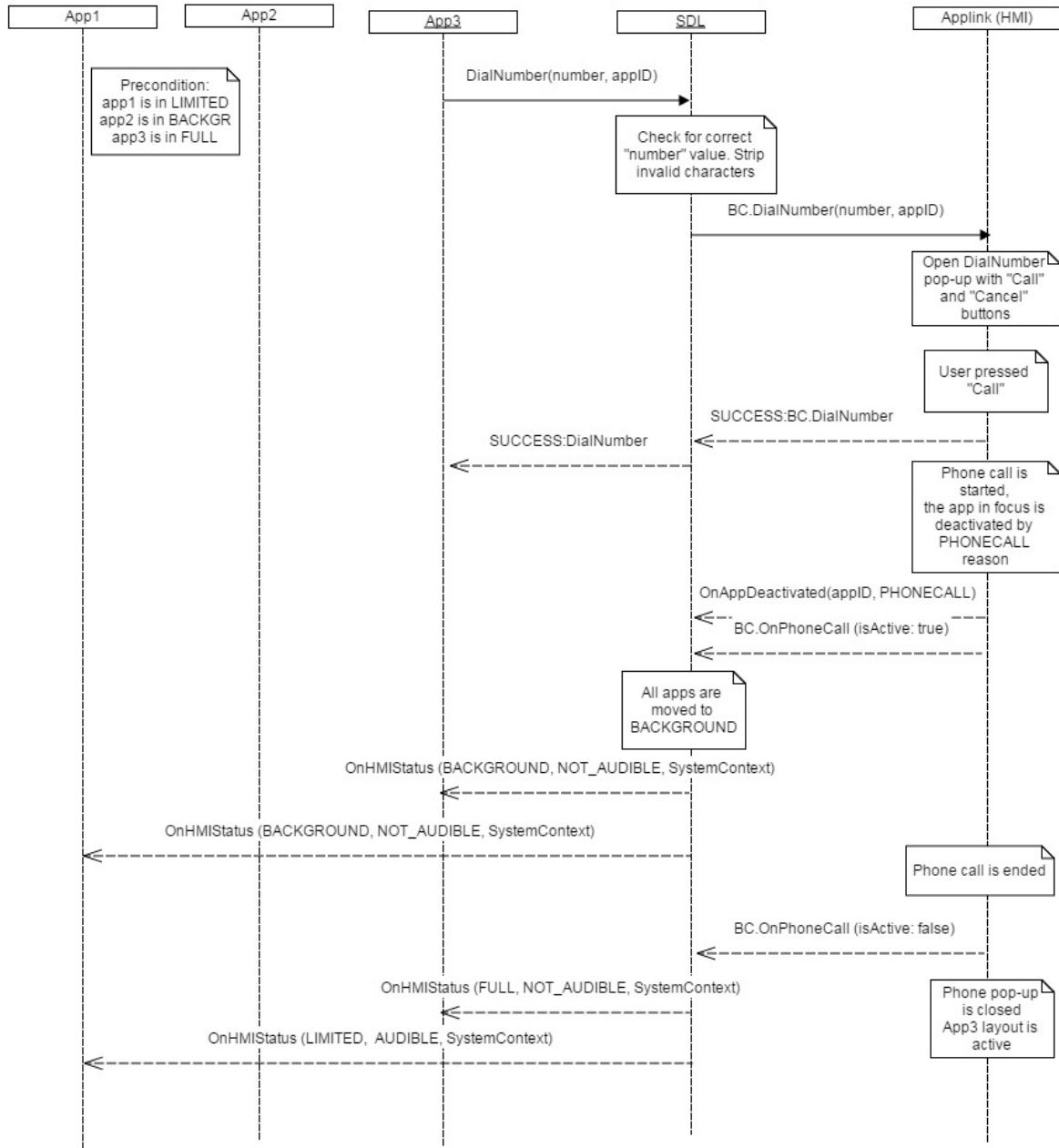
- 1) Send notification with appropriate parameter value when active call on HMI has been started/ended
- 2) Change internally on HMI HMILevels of the apps in FULL and-or LIMITED to BACKGROUND after phone call started
- 3) Resume the applications to the previous-to-phonecall state on HMI (as SDL does not send BC.ActivateApp or BC.OnResumeAudioSource to HMI after the phone call is ended)

6.26.1.1 Parameters

Param Name	Type	Mandatory	Additional	Description
isActive	Boolean	true		Must be 'true' - when a phone call is started on HMI. Must be 'false' when the phone call is ended on HMI

6.26.2 Sequence Diagrams

6.26.2.1 OnPhoneCall



6.26.3 JSON Messages Examples

```
{
  "jsonrpc" : "2.0",
  "method" :
"BasicCommunication.OnPhoneCall"
  "params" :
  {
    "isActive": true
  }
}
```

6.27 OnEmergencyEvent

6.27.1 Description

Type:	Notification
Sender:	HMI->SDL
Purpose:	For SDL to change the audioStreamingState of the related apps to NOT_AUDIBLE when "enabled:true" and back to AUDIBLE when "enabled:false"

Safety Feature is HMI's specific mode when " «Rear view camera are active modes. The main idea from SDL<->HMI point of view is that navigation/audio streamming mustn't interfere with Rear Camera View mode. HMI is responsible for managing audio/video data during "911"/"RearCamera" modes.

Note: When receive OnEmergencyEvent(enabled:true), SDL moves all apps with AudioStreamingState AUDIBLE to NOT_AUDIBLE state and return to previous state when get OnEmergencyEvent(enabled:false),

HMI must:

- 1) send OnEmergencyEvent when «Rear view camera" become active mode
- 2) reject navigation requests request since the "Rear view camera" is active
- 3) govern "Rear view camera" and how it coexists with SDL

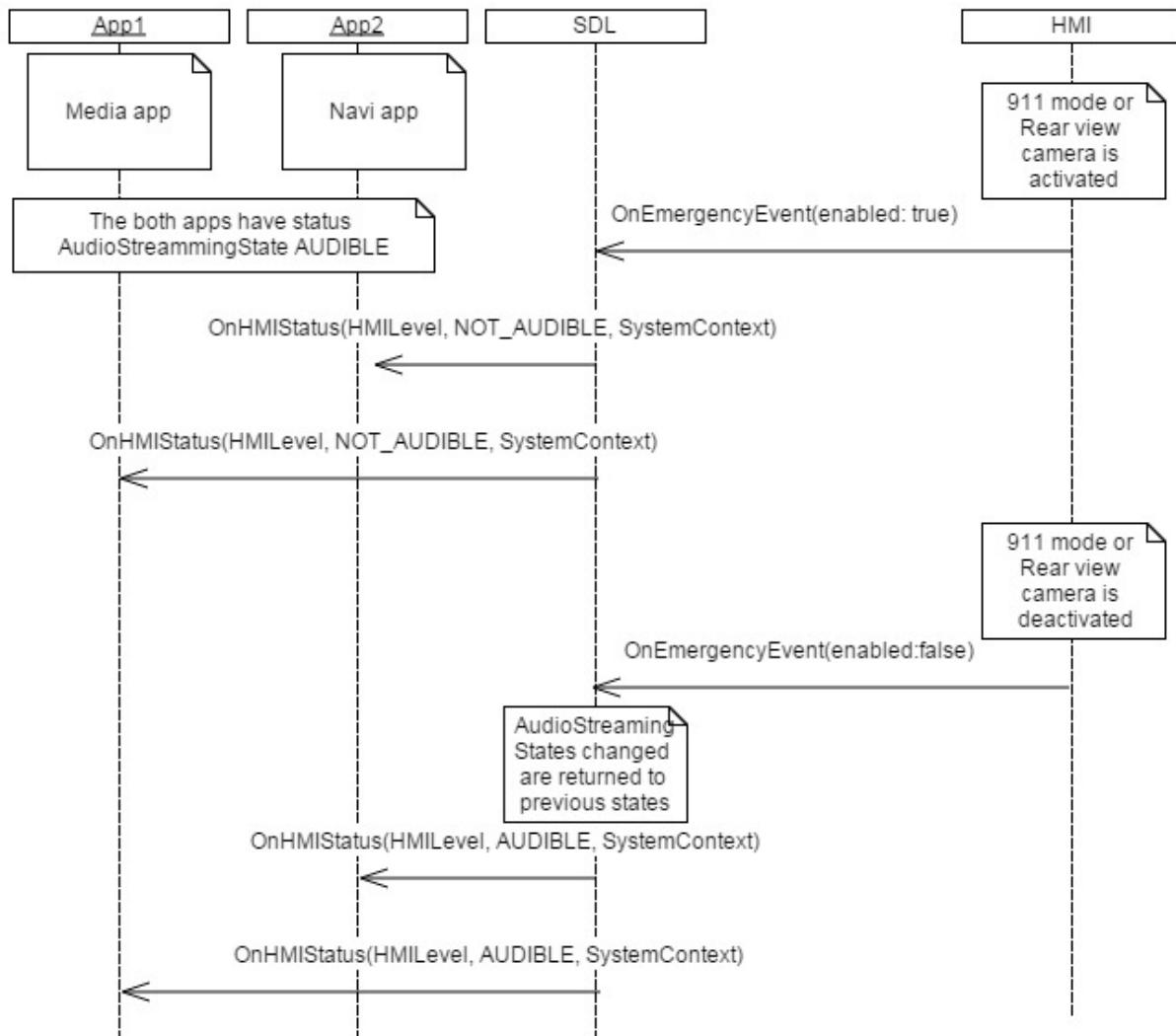
Note: SDL ignores all invalid notifications which come from HMI (invalid JSON, invalid data types/bounds etc)

6.27.1.1 Parameters

Param Name	Type	Mandatory	Additional	Description
enabled	Boolean	true		Defines if the "911 Assist" mode or rear view camera has been activated on HMI

6.27.2 Sequence Diagrams

6.27.2.1 OnEmergencyEvent



6.27.3 JSON Messages Examples

```
{
    "jsonrpc" : "2.0",
    "method" : "BasicCommunication.
OnEmergencyEvent"
    "params" :
    {
        "enabled" : "true"
    }
}
```

6.28 OnAwakeSDL

6.28.1 Description

Type:	Notification
Sender:	HMI->SDL
Purpose:	To notify SDL to return to normal operating after 'Suspend' event

The notification is sent to SDL to notify it there won't be IGNITION_OFF which had to go after OnExitAllApplications(SUSPEND) notification which had been sent before by HMI.

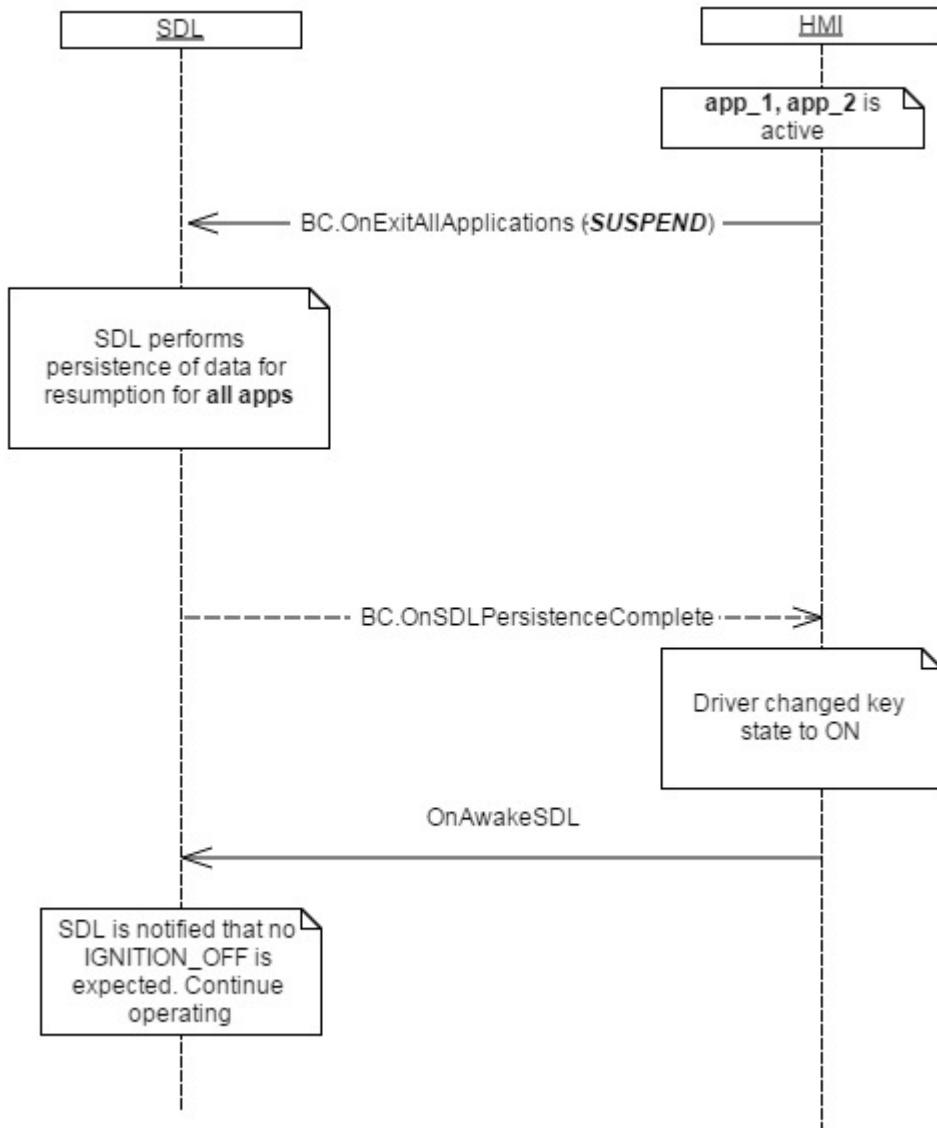
Note: SDL ignores all invalid notifications which come from HMI (invalid JSON, invalid data types/bounds etc)

HMI must:

- 1) Send OnAwakeSDL on ACC key state not being changed during the door is opened and closed for 2 times
- 2) Send OnAwakeSDL on ACC key position is changed to ON state

6.28.2 Sequence Diagrams

6.28.2.1 OnAwakeSDL



6.28.3 JSON Messages Examples

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

6.29 DialNumber

6.29.1 Description

Type:	Function
Sender:	SDL
Purpose:	SDL initiates a call to a specific phone

	number.
--	---------

6.29.2 Request

6.29.2.1 Behavior

The request is considered to be executed successfully in case the user presses "Call" button on DialNumber phone dialog. **SDL makes check** for correct phone number on it's side and transfers stripped number to HMI. SDL performs the following checks :

- 1) Strip any characters except of 0-9 and * # , ;+
- 2) Return INVALID_DATA to mobile side without transferring to HMI in case characters "/\n", "/\t", **the space/s** are a part of the number or as a result of strip the "number" param became empty.

HMI must:

- 1) Show Dial Number pop-up on HMI with 2 buttons "Call" and "Cancel"
- 2) Send OnAppDeactivated(PHONECALL) notification to SDL when the active phone call has been started on HMI. The notification must be sent to all applications which have active audio source on HMI
- 3) Send BC.OnOnPhoneCall(isActive:true) notification to SDL when the active phone call has been started on HMI
- 4) Send BC.OnOnPhoneCall(isActive:false) notification to SDL when the active phone call has been ended on HMI
- 5) Always respond on BC.DialNumber back with one of the Results ([see 6.29.3](#))

Note: In case HMI doesn't return the response to BasicCommunication.DialNumber, mobile side will never get an answer from SDL. SDL's default timeout response isn't applicable for DialNumber mobile API.

6.29.2.1 Parameters

Param Name	Type	Mandatory	Additional	Description
number	String	true	maxlength = 40	The number to dial. All characters shall be stripped from string by SDL except digits 0-9 and * # , ;+
appID	Integer	true		ID of application that initiates the call

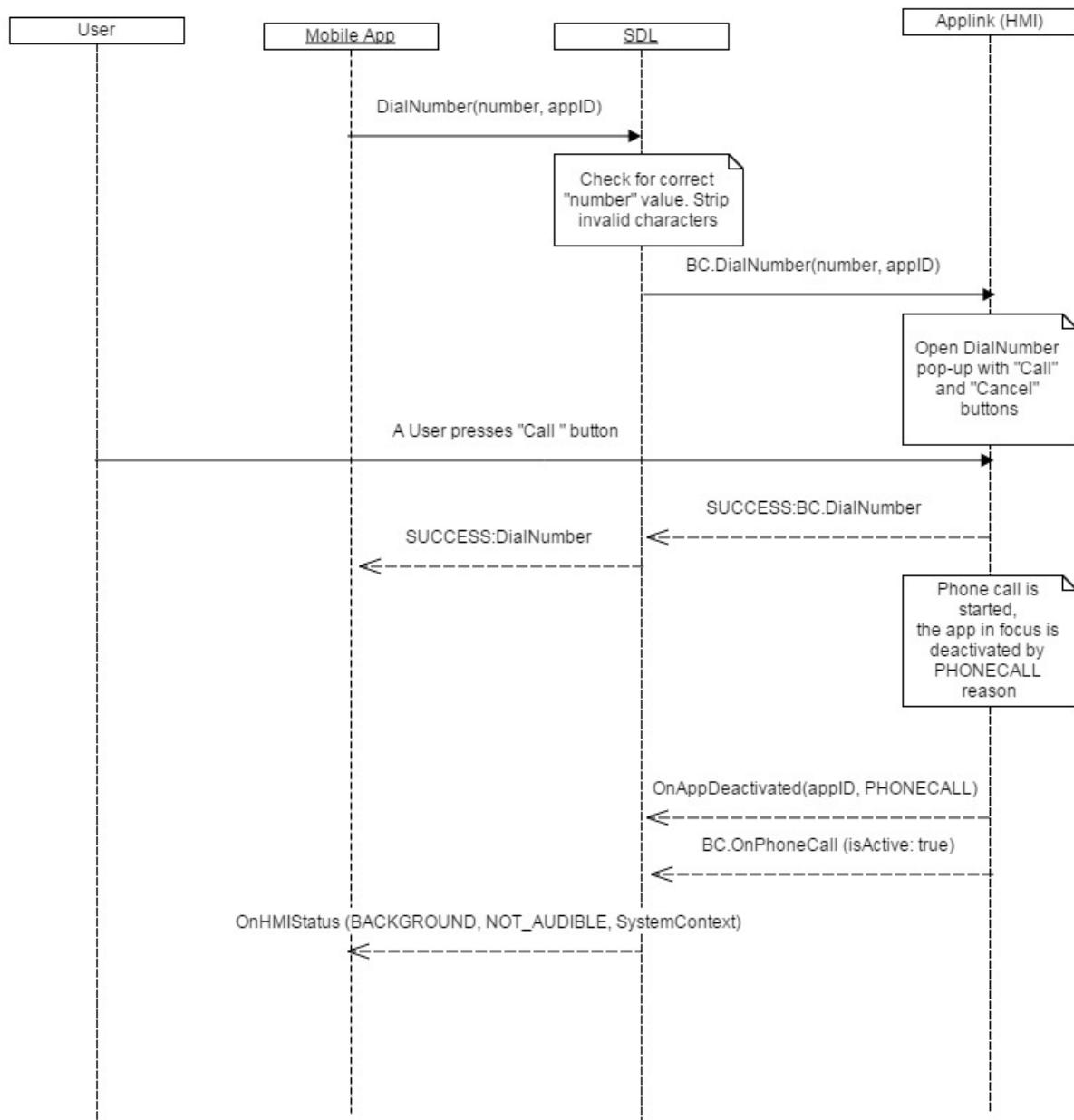
6.29.3 Response

Result	Description	Message type	Mess	Notes
WebSo	D-	age		

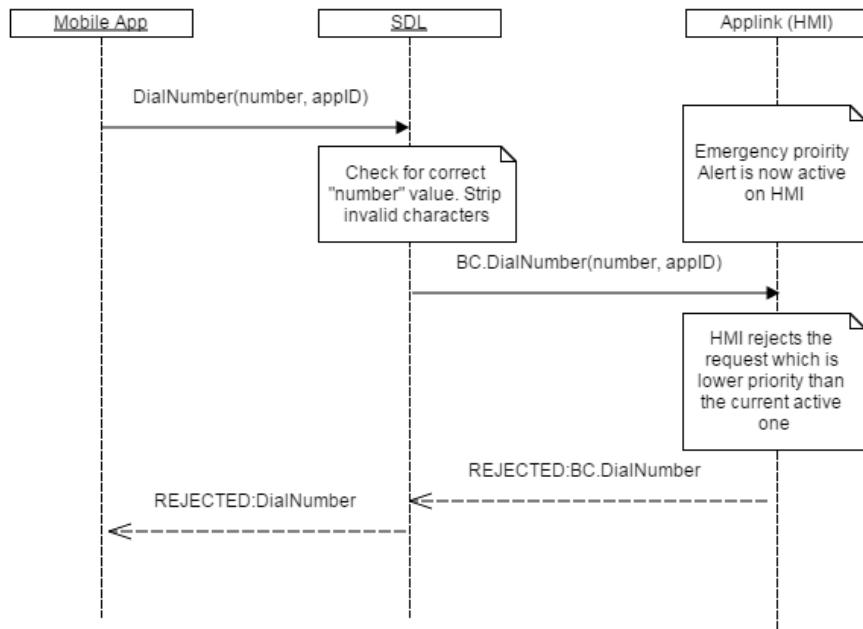
		cket	Bus	Para ms	
Success	SUCCESS Phone call started on HMI (he user presses 'call' button)	JSON response	Regul ar respo nse	code: 0	Reques t execut ed success fully
Failure	REJECTED 2) HMI is busy with higher priority RPC	JSON response	Regul ar respo nse	code: 4	
	TIMED_OUT when the DialNumber dialog is closed by HMI-defined default timeout.			code: 10	
	ABORTED a) when the user has not yet done his choice and some higher-priority event interrupted the DialNumber dialog b) when the user presses 'cancel' button			code: 5	
	GENERIC_ERROR When anything else unpredictable happened on HMI (except of other codes)			code: 22	

6.29.4 Sequence Diagrams

6.29.5.1 DialNumber Success



6.29.5.1 DialNumber Failed



6.29.5 JSON Messages Examples

6.29.5.1 Request

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

6.29.5.2 Response

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

6.29.5.3 Error message

```
{  
    "id" : 59,  
    "jsonrpc" : "2.0",  
    "error" :  
    {  
        "code" : -32603,  
        "message" : "Method not found"  
    }  
}
```

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

7 UI Component Description

7.1 IsReady

7.1.1 Description

Type:	Function
Sender:	SDL
Purpose:	Know if UI module is ready.

The request comes after HMI's readiness is confirmed via [OnReady](#) notification. SDL requires the information about whether the display is physically present on HU and if so whether it can be involved in application activities.

Note:

If HMI responded that UI module is unavailable, SDL will not further send the requests related to UI interface.

7.1.2 Request

7.1.2.1 Behavior

HMI must:

- 1) Check whether UI module is present and ready
- 2) Respond correspondingly to results of this check.

7.1.3 Response

Note:

There is a difference in message type for WebSocket and D-Bus connection (see the table below).

Resu lt	Description	Message type		Messag e Params	Not es
		WebSo cket	D- Bus		
Success	SUCCESS HMI has the information about UI availability.	JSON respons e	Regul ar respo nse	availa ble, Code: 0	See the tabl e belo w.
Failure	GENERIC_E RROR HMI doesn't have the	JSON respons e	Regul ar respo nse	Code: 22	.

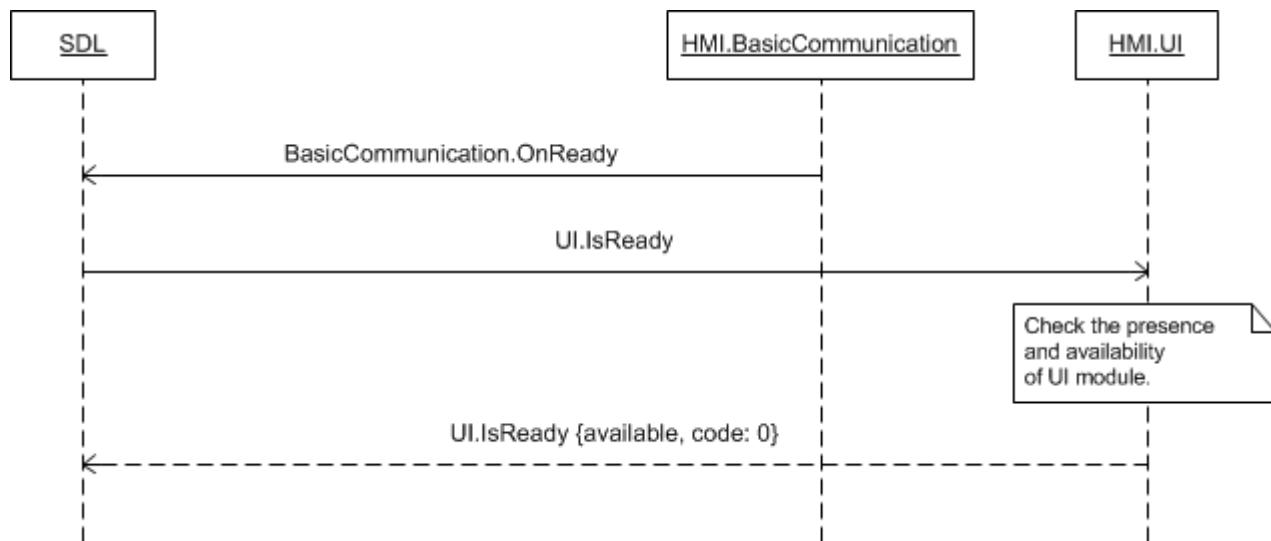
	information about UI availability or some failure occurred				
--	--	--	--	--	--

7.1.3.1 Parameters

Param Name	Type	Mandatory	Description
availabe	Boolean	true	Must be - 'true' if UI is present and ready - 'false' if not.

7.1.4 Sequence Diagrams

7.1.4.1 UI.IsReady and preceding OnReady



7.1.5 JSON Messages Examples

7.1.5.1 Request

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

7.1.5.2 Response

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

```
}
```

7.1.5.3 Error message

```
{
    "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"
        }
    }
}
```

7.2 GetCapabilities

7.2.1 Description

Type:	Function
Sender:	SDL
Purpose:	Learn the UI capabilities of HU.

Once UI module is confirmed to be ready (via response to [IsReady](#) RPC), SDL starts discovering its capabilities via [GetCapabilities](#) and [GetSupportedLanguages](#) RPCs.

The response to [UI.GetCapabilities](#) is assumed to bring all the information required for correct configuring the 'display' requests to UI.

7.2.2 Request

7.2.2.1 Behavior

HMI must:

- 1) Check the capabilities of:
 - Display: its type, the fields supported, the format of media clock supported, whether displaying images are supported and of what type if so, image capabilities, available persistent display templates and number of presets etc
 - Zone the UI is located in (front/back display).
 - Soft buttons: whether the notifications on events of softbutton depress/release and on button presses of short/long are supported.
 - Embedded Navigation supported
 - Phone calls supported
 - Audio capturing: the sampling rate, bits per sample, audio type supported.

- 2) Respond correspondingly to results of this check.

Note:

The expected UI capabilities are described in the section [7.2.3.1 Parameters](#) linked to the corresponding structures and enumerations with more detailed information.

7.2.3 Response

Note:

There is a difference in message type for WebSocket and D-Bus connection (see the table below).

Result	Description	Message type		Message Params	Notes
		WebSocket	D-Bus		
Success	SUCCESS HMI has the information about UI capabilities.	JSON response	Regular response	displayCapabilities, hmiZoneCapabilities, softButtonCapabilities, audioPassThruCapabilities, Code: 0	See section 7.2.3.1 Parameters .
Failure	GENERIC_ERROR Some failure occurred	JSON error message	Regular response	Code: 22	.

7.2.3.1 Parameters

Param Name	Type	Mandatory	Additional	Description
displayCapabilities	Common.Display Capabilities	true		The capabilities of HMI's display: its type, supported textfields, whether the graphics displaying is supported, the supported formats of media clock. See DisplayCapabilities .
hmiCapabilities	HMICapabilities	false		Specifies the HMI capabilities of navigation and phonecall support. See HMICapabilities
softButtonCapabilities	Common.SoftButtonCapabilities	false	Array = true minsize = 1 maxsize = 100	Must be returned if the platform supports on-screen soft buttons. Contains the soft buttons capabilities: whether the up/down events, long/short press, referencing image are supported. See SoftButtonCapabilities .
hmiZoneCapabilities	Common.HmiZone Capabilities	true		Specifies HMI Zones in the vehicle (front/back). See HmiZoneCapabilities .
audioPassThruCapabilities	Common.AudioPassThruCapabilities	true		Specifies the capabilities of audio capturing: sampling rate, bits per sample, audio type. See AudioPassThruCapabilities .

7.2.3.2 DisplayCapabilities Structure

Param Name	Type	Mandatory	Additional	Description
displayType	Common.DisplayType	true	-	The type of the display that is installed on HU. See DisplayType .
textFields	Common.TextField	true	Array = true minsize = 0 maxsize = 100	A set of all fields that support displaying text data. If there are no textfields supported HMI must send the empty array. See TextField .
imageFields	Common.ImageField	false	Array = true minsize = 1 maxsize = 100	A set of all fields that support images. See ImageField
mediaClockFormats	Common.MediaClockFormat	true	Array = true minsize = 1 maxsize = 100	A set of all supported formats of the media clock. See MediaClockFormat .
imageCapabilities	Common.ImageType	false	Array = true minsize = 0 maxsize = 2	The array of supported image types (static and/or dynamic). The empty array should be returned if the platform does not support displaying images. See ImageType .
graphicSupported	Boolean	true	-	Must be: - 'true' if display's persistent screen supports referencing a static or dynamic image. - 'false' if not.
templatesAvailable	String	true	Array = true minsize = 0 maxsize = 100 maxlength = 100	A set of all predefined persistent display templates available on headunit. To be referenced in SetDisplayLayout.
screenParams	Common.ScreenParams	false	-	A set of all parameters related to a prescribed screen area (e.g. for video / touch input).
numCustomPresetsAvailable	Integer	false	minvalue = 1 maxvalue = 100	

[Back to Parameters](#)

7.2.3.3 DisplayType Enumeration

Element name	Short Description
CID	Center Information Display. This display type provides a 2-line x 20 character "dot matrix" display.
TYPE2	TYPE II display. 1 line older radio head unit.

Element name	Short Description
TYPE5	TYPE V display Old radio head unit.
NGN	Next Generation Navigation display.
GEN2_8_DMA	GEN-2, 8 inch display.
GEN2_6_DMA	GEN-2, 6 inch display.
MFD3	3 inch GEN1.1 display
MFD4	4 inch GEN1.1 display
MFD5	5 inch GEN1.1 display
GEN3_8-INCH	GEN-3, 8 inch display.

[Back to DisplayCapabilities](#)

7.2.3.4 TextFieldName Enumeration

Element name	Short Description
mainField1	The text that must be displayed in a single or upper display line. If this value is not set, the text of mainField1 must stay unchanged. If this text is empty "", the field must be cleared. Applies to Show, section 7.5 .
mainField2	The text that must be displayed on the second display line. If this text is not set, the text of mainField2 must stay unchanged. If this text is empty "", the field must be cleared. Applies to Show, section 7.5 .
mainField3	The text that must be displayed on the second "page" first display line. If this text is not set, the text of mainField3 must stay unchanged. If this text is empty "", the field must be cleared. Applies to Show, section 7.5 .
mainField4	The text that must be displayed on the second "page" second display line. If this text is not set, the text of mainField4 must stay unchanged. If this text is empty "",

Element name	Short Description
	<p>the field must be cleared.</p> <p>Applies to Show, section 7.5.</p>
statusBar	<p>The text is placed in the status bar area.</p> <p>Note: This relates to navigation displays</p> <p>If this parameter is omitted, the status bar text must remain unchanged.</p> <p>If this parameter is an empty string, the field must be cleared.</p> <p>If provided and the display has no status bar, this parameter must be ignored.</p> <p>Applies to Show, section 7.5.</p>
mediaClock	<p>Text value for MediaClock field. Shall arrive in the form as described in the MediaClockFormat enumeration</p> <p>If this text is set, any automatic media clock updates previously set with SetMediaClockTimer must be stopped.</p> <p>Applies to Show, section 7.5.</p>
mediaTrack	<p>The text that should be displayed in the track field. This field should be valid only for media applications on.</p> <p>If this text is not set, the text of mediaTrack must stay unchanged.</p> <p>If this text is empty "", the field must be cleared.</p> <p>Applies to Show, section 7.5.</p>
alertText1	<p>The text that must be displayed in the top field of the display during the Alert.</p> <p>Applies to Alert, section 7.5.</p>
alertText2	<p>The text that must be displayed in the bottom field of the display during the Alert.</p>

Element name	Short Description
	Applies to Alert, section 7.5.
alertText3	The optional third line of the alert text field. Applies to Alert, section 7.5.
scrollableMessageBody	The long body of text layout that can include newlines and tabs. Applies to ScrollableMessage, section 7.18.
initialInteractionText	Must be displayed when the interaction begins. The text must be displayed on the first line of a multiline display, and must be centered. Applies to PerformInteraction, section 7.11.
navigationText1	The text that must be displayed on the first line of navigation text. Applies to ShowConstantTBT, section 12.3.
navigationText2	The text that must be displayed on the second line of navigation text. Applies to ShowConstantTBT, section 12.3.
ETA	Estimated Time of Arrival for navigation. Applies to ShowConstantTBT, section 12.3.
totalDistance	Total distance to destination for navigation. Applies to ShowConstantTBT, section 12.3.
navigationText	Navigation text for UpdateTurnList. Applies to Turn structure of UpdateTurnList, section 12.4.
audioPassThruDisplayTe xt1	The first line of text that must be displayed during audio capture. Applies to PerformAudioPassTh ru, section 7.19.

Element name	Short Description
audioPassThruDisplayText2	The second line of text that must be displayed during audio capture. Applies to PerformAudioPassThru, section 7.19 .
sliderHeader	The text that must be displayed on the header of slider. Applies to Slider, section 7.17 .
sliderFooter	The text that must be displayed on the footer of slider. Applies to Slider, section 7.17 .
notificationText	Text of the notification to be displayed on screen. Currently not used.
menuName	Primary text for Choice. Applies to PerformInteraction, section 7.11 .
secondaryText	Secondary text for Choice. Applies to PerformInteraction, section 7.11 .
tertiaryText	Tertiary text for Choice. Applies to PerformInteraction, section 7.11 .
menuTitle	Optional text to label an app menu button (for certain touchscreen platforms).
timeToDestination	The line to display the time to destination. Applies to ShowConstantTBT, section 12.3 .
turnText	Currently not used.

[Back to DisplayCapabilities](#)

7.2.3.5 ImageField Structure

Param Name	Type	Mandatory	Additional	Description
name	Common.ImageFileDialogName	true	-	The name that identifies the field. See ImageFieldName.
imageTypeSupported	Common.FileType	false	array = true minsize = 1 maxsize = 100	The image types that are supported in this field. See FileType.
imageResolution	Common.ImageResolution	false	-	The image resolution of this field.

[Back to DisplayCapabilities](#)

7.2.3.6 ImageFieldName Enumeration

Element name	Value	Short Description
softButtonImage	0	The image field for SoftButton
choiceImage	1	The first image field for Choice
choiceSecondaryImage	2	The secondary image field for Choice
vrHelpItem	3	The image field for vrHelpItem
turnIcon	4	The image field for Turn
menuIcon	5	The image field for the menu icon in SetGlobalProperties
cmdIcon	6	The image field for AddCommand
appIcon	7	The image field for the app icon (set by setAppIcon)
graphic	8	The image field for Show
showConstantTBTIcon	9	The primary image field for ShowConstantTBT
showConstantTBTNextTurnIcon	10	The secondary image field for ShowConstantTBT
locationImage	11	The optional image of a destination / location

7.2.3.7 FileType Enumeration

Element name	Value	Short Description
GRAPHIC_BMP	0	
GRAPHIC_JPEG	1	
GRAPHIC_PNG	2	
AUDIO_WAVE	3	
AUDIO_MP3	4	
AUDIO_AAC	5	
BINARY	6	
JSON	7	

7.2.3.8 ImageResolution Structure

Param Name	Type	Mandatory	Additional	Description
resolutionWidth	Integer	true	minvalue = 1 maxvalue = 10000	The image resolution width.
resolutionHeight	Integer	true	minvalue = 1 maxvalue = 10000	The image resolution height.

7.2.3.9 MediaClockFormat Enumeration

Element name	Short Description
CLOCK1	<pre>minutesFieldWidth = 2; minutesFieldMax = 19; secondsFieldWidth = 2; secondsFieldMax = 99; maxHours = 19; maxMinutes = 59; maxSeconds = 59; Is used for Type II, NGN and CID head units.</pre>
CLOCK2	<pre>minutesFieldWidth = 3; minutesFieldMax = 199; secondsFieldWidth = 2; secondsFieldMax = 99; maxHours = 59; maxMinutes = 59; maxSeconds = 59; Is used for Type V head units.</pre>
CLOCK3	<pre>minutesFieldWidth = 2; minutesFieldMax = 59; secondsFieldWidth = 2; secondsFieldMax = 59; maxHours = 9; maxMinutes = 59; maxSeconds = 59; Is used for GEN1.1 (i.e. MFD3/4/5) head units.</pre>
CLOCKTEXT1	<p>5 characters possible</p> <p>Format: 1 sp c : sp c c 1 sp : digit "1" or space c : character out of following character set: sp 0-9 [letters : sp : colon or space</p> <p>Is used for Type II head unit</p>
CLOCKTEXT2	<p>5 chars possible</p> <p>Format: 1 sp c : sp c c 1 sp : digit "1" or space c : character out of following character set: sp 0-9 [letters : sp : colon or space</p>

Element name	Short Description
	Is used for CID and NGN head unit.
CLOCKTEXT3	6 chars possible Format: 1 sp c c : sp c c 1 sp : digit "1" or space c : character out of following character set: sp 0-9 [letters] : sp : colon or space Is used for Type V head unit.
CLOCKTEXT4	6 chars possible Format: c : sp c c : c c : sp : colon or space c : character out of following character set: sp 0-9 [letters]. Is used for GEN1.1 (i.e. MFD3/4/5) head units.

[Back to DisplayCapabilities](#)

7.2.3.10 ImageType Enumeration

Element name	Short Description
STATIC	Static image. The image that is sent as the binary or hex code within the request.
DYNAMIC	Dynamic image. The image that is stored on HMI and just a link to it is further used within requests.

[Back to DisplayCapabilities](#)

7.2.3.11 ScreenParams Structure

Param Name	Type	Mandatory	Description
resolution	Common.ImageResolution	true	The resolution of the prescribed screen area. See ImageResolution .
touchEventAvailable	Common.TouchEventCapabilities	false	Types of screen touch events available in screen area.

7.2.3.12 TouchEventCapabilities Structure

Param Name	Type	Mandatory	Description
pressAvailable	Boolean	true	
multiTouchAvailable	Boolean	true	
doublePressAvailable	Boolean	true	

7.2.3.13 SoftButtonCapabilities Structure

Param Name	Type	Mandatory	Description

Param Name	Type	Mandatory	Description
shortPressAvailable	Boolean	true	Must be - 'true' if soft buttons support a short press - 'false' if not. See ButtonPress Mode for more information.
longPressAvailable	Boolean	true	Must be - 'true' if soft buttons support a LONG press - 'false' if not. See ButtonPress Mode for more information.
upDownAvailable	Boolean	true	Must be - 'true' if soft buttons support "button down" and "button up". - 'false' if not. See ButtonEvent Mode for more information.
imageSupported	Boolean	true	Must be - 'true' if soft buttons support referencing image - 'false' if not.

[Back to Parameters](#)

7.2.3.14 HmiZoneCapabilities Enumeration

Element name	Short Description
FRONT	Indicates UI display available to front seat passengers.
BACK	Indicates UI display available to rear seat passengers.

[Back to Parameters](#)

7.2.3.15 AudioPassThruCapabilities Structure

Param Name	Type	Mandatory	Description
samplingRate	Common.SamplingRate	true	The number of samples per second. Defines the sampling rate supported by the HU for audio capturing (e.g. 8 kHz, 16 kHz, etc.). See SamplingRate .
bitsPerSample	Common.BitsPerSample	true	The number of bits of information in each sample. Defines the quality of audio capturing supported by HU. See BitsPerSample .

audioType	Common.AudioType	true	Specifies the type of audio data being requested. See AudioType .
-----------	------------------	------	--

[Back to Parameters](#)

7.2.3.16 SamplingRate Enumeration

Element name	Short Description
8KHZ	Sampling rate of 8 kHz
16KHZ	Sampling rate of 16 kHz
22KHZ	Sampling rate of 22 kHz
44KHZ	Sampling rate of 44 kHz

[Back to AudioPassThruCapabilities](#)

7.2.3.17 BitsPerSample Enumeration

Element name	Short Description
8_BIT	8 bit per sample
16_BIT	16 bit per sample

[Back to AudioPassThruCapabilities](#)

7.2.3.18 AudioType Enumeration

Element name	Short Description
PCM	Pulse Code Modulated audio.

[Back to AudioPassThruCapabilities](#)

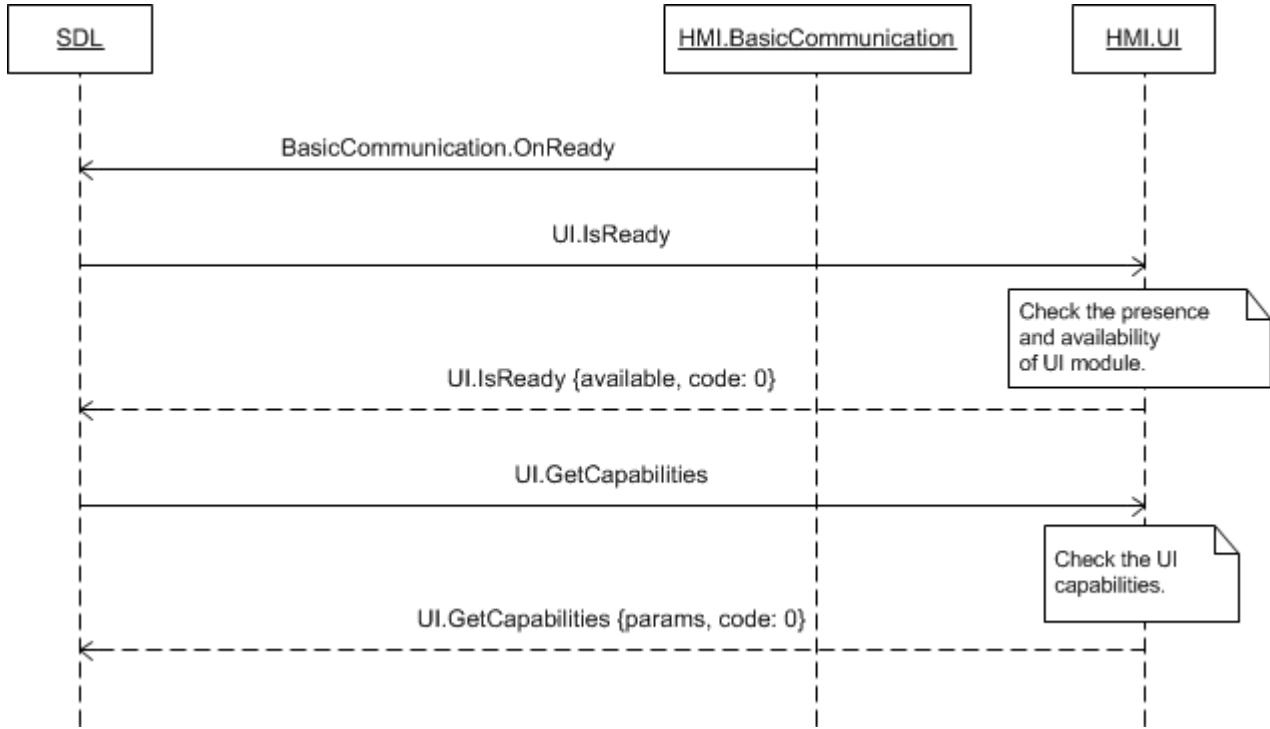
7.2.3.19 HMICapabilities structure

Param Name	Type	Mandatory	Additional	Description
navigation	boolean	false		Availability of built-in Navigation. True: available, False: not available
phoneCall	boolean	false		Availability of built-in phone. True: available, False: not available

[Back to Parameters](#)

7.2.4 Sequence Diagrams

7.2.4.1 UI.GetCapabilities and preceding IsReady



7.2.5 JSON Messages Examples

7.2.5.1 Request

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

7.2.5.2 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
        },
        "softButtonCapabilities" :
{
            "shortPressAvailable" :
true,
            "longPressAvailable" :
true,
            "upDownAvailable" : true,
            "imageSupported" : true
        },
        "hmiZoneCapabilities" : FRONT,
        "audioPassThruCapabilities" :
{
            "samplingRate" : 44KHZ,
            "bitsPerSample" : 8_BIT,
            "audioType" : PCM
        },
        "code" : 0,
        "method" : "UI.GetCapabilities"
    }
}

```

7.2.5.3 Error message

```

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

```

7.3 GetSupportedLanguages

7.3.1 Description

Type:	Function
-------	----------

Sender:	SDL
Purpose:	Get the UI supported languages.

Once UI module is confirmed to be ready (via response to [IsReady](#) RPC) SDL starts discovering its capabilities via [GetCapabilities](#) and [GetSupportedLanguages](#) RPCs.

Response to `UI.GetSupportedLanguages` is assumed to bring the information about what languages are supported for displaying the text information on UI. Having obtained this information SDL will monitor the language parameter within RPCs from mobile application(s) and reject the requests containing language not supported by HMI.

Note:

The list of languages recognized by SDL is provided in the section [7.3.3.2 Language Enumeration](#).

7.3.2 Request

7.3.2.1 Behavior

HMI must:

- 1) Check the UI supported languages
- 2) Respond with the list of supported languages

7.3.3 Response

Note:

There is a difference in message type for WebSocket and D-Bus connection (see the table below).

Result	Description	Message type		Message Params	Notes
		WebSocket	D-Bus		
Success	SUCCESS HMI has the information about UI supported languages.	JSON response	Regular response	languages, code: 0	See the table below.
Failure	DATA_NOT_AVAILABLE The information about UI supported languages is not available.	JSON error message	Regular response	code: 9	Applicable for this RPC result codes. Please see Result Enumeration for all SDL-
	INVALID_DATA The data sent is invalid (invalid JSON syntax)			code: 11	

	GENERIC_ERROR The unknown issue occurred or other codes are not applicable.			code: 22	supported codes.
--	--	--	--	-------------	------------------

7.3.3.1 Parameters

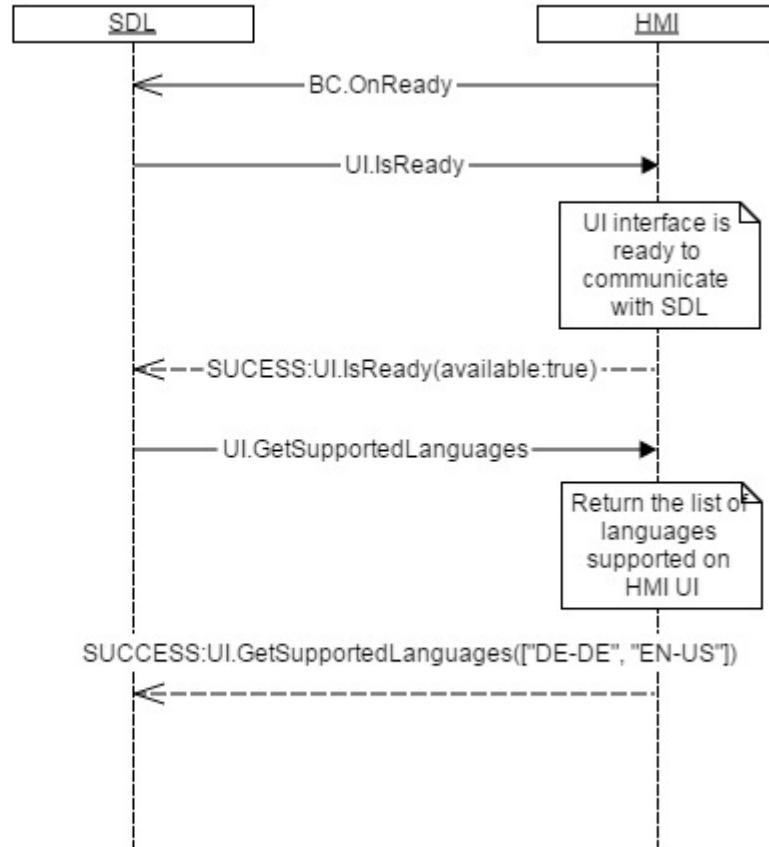
Param Name	Type	Mandatory	Additional	Description
languages	Common.Lang uage	true	Array = true minsize = 1 maxsize = 100	List of UI supported languages. See Language.

7.3.3.2 Language Enumeration

Element name	Short Description
AR-SA	Arabic – Saudi Arabia
CS-CZ	Czech – Czech Republic
DA-DK	Danish – Denmark
DE-DE	German – Germany
EN-AU	English – Australia
EN-GB	English – GB
EN-US	English – US
ES-ES	Spanish – Spain
ES-MX	Spanish – Mexico
FR-CA	French – Canada
FR-FR	French – France
IT-IT	Italian – Italy
JA-JP	Japanese – Japan
KO-KR	Korean – South Korea
NL-NL	Dutch (Standard) – Netherlands
NO-NO	Norwegian - Norway
PL-PL	Polish – Poland
PT-PT	Portuguese – Portugal
PT-BR	Portuguese – Brazil
RU-RU	Russian - Russia
SV-SE	Swedish – Sweden
TR-TR	Turkish – Turkey
ZH-CN	Mandarin – China
ZH-TW	Mandarin – Taiwan
NL-BE	Dutch Belgium (Flemish)
EL-GR	Greek
HU-HU	Hungarian
FI-FI	Finnish
SK-SK	Slovak

7.3.4 Sequence Diagrams

7.1.4.1 GetSupportedLanguages with preceding IsReady



7.3.5 JSON Messages Examples

7.3.5.1 Request

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

7.3.5.2 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"  
    }  
}
```

```
}
```

7.3.5.3 Error message

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

7.4 Alert

7.4.1 Description

Type:	Function
Sender:	SDL
Purpose:	Display alert message on HMI.

UI.Alert request notify the user by some readable information which requires to be shown on a display. It's expected to be a pop-up with some text, sent by the mobile application. SDL transfers the text to HMI via UI.Alert request.

Note:

SDL uses the information received via response to [UI.GetCapabilities](#) for sending only the data which corresponds to supported text fields names or other supported entities on HMI like images for soft buttons once confirmed to be supported by HM, etc.).

Together with UI.Alert method SDL may also send:

- BasicCommunication.PlayTone for turning the User's attention to the Alert message with the help of HMI standard notifying sound
- TTS.Speak to provide an audible part of Alert to the User.

There're 2 cases of using Alert on HMI. An application may send an Alert message with or without TTS portion (TTSCunks):

1) In case TTS portion is present in app's Alert ->
SDL->HMI: UI.Alert (params, alertType: BOTH)
SDL->HMI: TTS.Speak

2) In case TTS portion is omitted in app's Alert ->
SDL->HMI: UI.Alert (params, alertType: UI)

More details about around Alert messaging please find in section [7.4.4 Sequence Diagrams](#).

Note:

If Alert request comes together with PlayTone and Speak RPCs, **HMI must:**

- At first play the notifying sound (see also [6.18 PlayTone](#))
- Then speak the requested text

HMI must respond to UI.Alert not earlier than TTS.Speak is received:

SDL->HMI: UI.Alert (params, alertType: BOTH)

SDL->HMI: TTS.Speak

HMI->SDL: UI.Alert_response

SDL->HMI: StopSpeaking

HMI->SDL: StopSpeaking_response

HMI->SDL: Speak_response

If UI.Alert (params, alertType: UI) is received, HMI may respond to UI.Alert at any time when applicable according to HMI workflows.

OnSystemContext related to successful Alert: 'appId' parameter optionality:

1. HMI ought to not send AppID for MENU and HMI_OBCSURED system contexts. Only the apps in FULL may get these updates.
2. In case HMI sends OnSystemContext of MENU and HMI_OBCSURED with appId by HMI, appId is ignored and anyway the notification is transferred to the app in FULL HMI Level
3. HMI must send OnSystemContext with appId parameter for MAIN and ALERT values. In case there's no such application with appId received, the notifications will be ignored by SDL

7.4.2 Request

7.4.2.1 Behavior

HMI must:

1. Notify about the Alert context via OnSystemContext notification (see the rules in 7.4.1).
2. HMI must respond UI.Alert earlier than SDL's default timeout of 10sec (applicable only to Alert without softButtons)
3. Display the alert dialog with:
 - The text information in the [named fields](#) (up to three fields may be requested)
 - Up to four soft buttons (optional)
 - Progress indicator showing the time to dialog closing.

Note:

The Alert dialog possible view is provided in the section [7.4.5 Possible Layout](#).

3. React on requested soft button(s) press with:

- Sending OnButtonPress/OnButtonEvent notifications

- Predefined behavior depending on the soft button type:
 - DEFAULT_ACTION type button press is assumed to close the alert dialog and return to the previously displayed screen
 - KEEP_CONTEXT type button press is assumed to renew the timeout for the alert dialog being displayed.
OnResetTimeout() must be sent by HMI in case of pressing a button with KEEP_CONTEXT
 - STEAL_FOCUS type button press is assumed to
 - Activate the application that has sent the Alert message while being not active on HMI
 - Return to the screen previously displayed if the application that sent the Alert RPC is active on HMI.

Note:

The diagrams describing the sequence of button press, notifications, messages is provided in section 7.4.4 Sequence Diagrams.

4. Display the alert dialog and then close it

- By the timeout requested by SDL
- Or upon the events:
 - One of the SDL-defined soft buttons press
 - HMI-defined ‘Close’/‘Back’/‘Return’ button press
 - VR activation
 - Another application activation or switching to any of HMI screens
 - Another RPC of a higher priority arrival.

5. Respond correspondingly after the definite event occurred (timeout, soft button press, etc.).

6. Notify via `OnSystemContext` about the system context is changed to main.

Note:

The detailed information about the events occurred and the applicable response codes are described in the section [7.4.3 Response](#).

HMI may:

Display the HMI-defined ‘Close’/‘Back’/‘Return’ button for providing the User with the possibility to dismiss the dialog. When such button is pressed

- The notification `OnButtonPress/OnButtonEvent` should not be sent to SDL
- The dialog must be closed
- The Alert RPC must be responded.

7.4.2.2 Parameters

Param Name	Type	Mandatory	Additional	Description
alertStrings	Common.TextFieldStruct	true	Array = true Minsize = 0 Maxsize = 3	Array of lines of alert text fields that must be displayed. See TextFieldStruct . Uses alertText1, alertText2, alertText3 from TextFieldName . If empty array is sent, the displayed alert message should not contain any text.
duration	Integer	true	Minvalue = 3000 Maxvalue = 10000	Timeout in milliseconds.
softButtons	Common.SoftButton	false	Array = true Minsize = 0 Maxsize = 4	Application-defined soft buttons. If omitted or the empty array is sent, the displayed alert message should not have any soft buttons. See section 5. Soft Buttons Behavior on HMI
progressIndicator	Boolean	false	-	If supported on the given platform, the alert dialog should include some sort of animation indicating that loading of a feature is progressing. e.g. a spinning wheel or hourglass, etc.
alertType	Common.AlertType	true	-	Defines if only UI or BOTH portions of the Alert request are being sent to HMI Side
appID	Integer	true	-	ID of the application related to this RPC.

7.4.2.3 TextFieldStruct Structure

Param Name	Type	Mandatory	Additional	Description
fieldName	Common.TextFieldName	true	-	The name of the field where the text must be displayed in.
fieldText	String	true	Maxlength = 500	The text to be displayed.

7.4.2.4 TextFieldName Enumeration

Only the text fields applicable to Alert RPC are described within this section. All the text fields names recognized by SDL are described in the section 13.1.14 [TextFieldName](#).

Element name	Short Description
alertText1	The first line in the top of the alert dialog for displaying the text.
alertText2	The second line of the alert dialog for displaying the text.

Element name	Short Description
alertText3	The third line of the alert dialog for displaying the text.

7.4.2.4 AlertType Enumeration

Element name	Short Description
UI	Alert RPC needs to be displayed only (no TTS portion).
BOTH	Alert RPC has TTS portion in addition to displayed pop-up.

7.4.3 Response

Note:

There is a difference in message type for WebSocket and D-Bus connection (see the table below).

Result	Description	Message type		Message Params	Notes
		WS	D-Bus		
Success	SUCCESS - The alert dialog has been displayed AND - Now is closed by one of the following reasons: <ul style="list-style-type: none">• The timeout.• The soft button with DEFAULT_ACTION system action press.• The soft button with STEAL_FOCUS system action press.	JSON response	Regular response	code : 0	-
	UNsupported_RESOURCE When images are sent by SDL but they aren't supported by HMI or HMI doesn't support the type of images sent (STATIC/DYNAMIC). The request is considered to be successfully executed anyway if the the request informed the user with Alert information.			code : 2	
Failure	ABORTED - The alert dialog has been displayed AND - Now is aborted by: <ul style="list-style-type: none">• HMI-defined 'Close' button press.• VR session.• Another application activation or switching to HMI screen• Another RPC of a higher priority.	JSON error message	Regular response	code : 5	Applicable for this RPC result codes. Please see Result Enumeration for all SDL-supported codes.

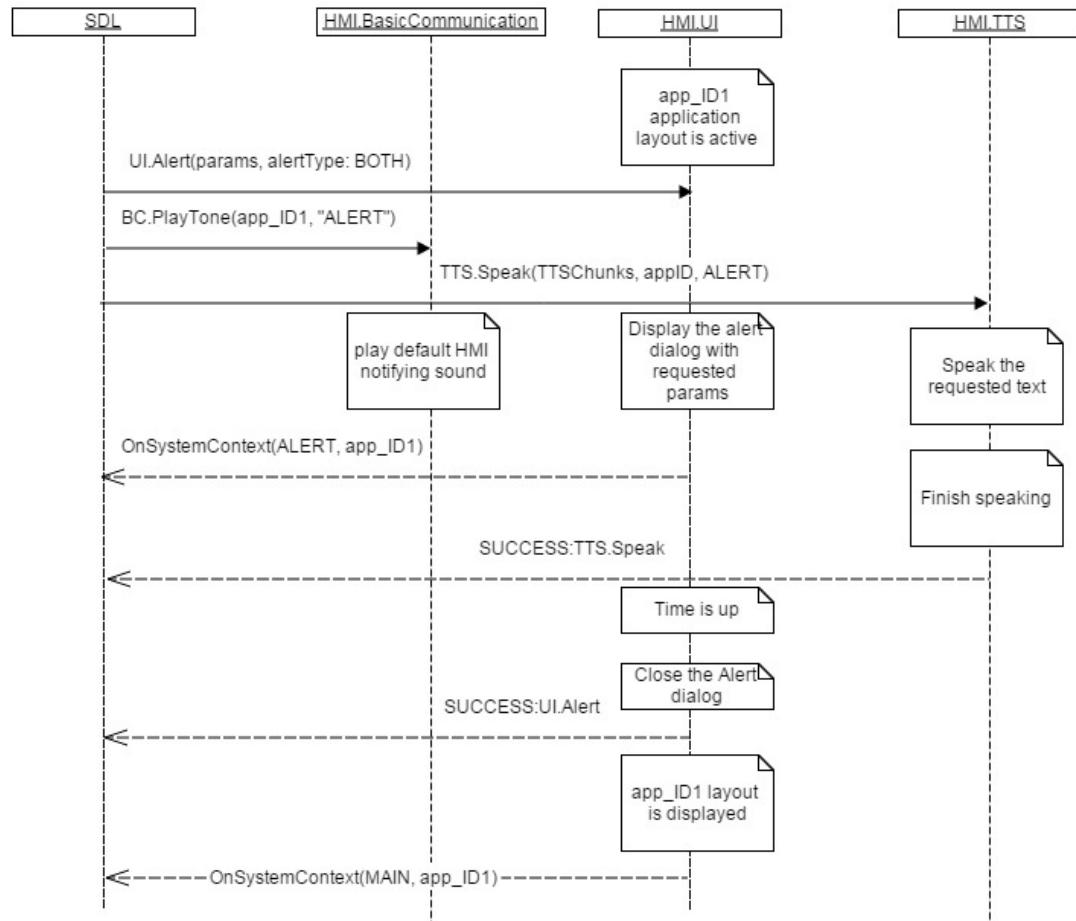
	REJECTED The Alert RPC is rejected because the higher priority RPC is now displayed on HMI.		tryAgainTime, code: 4	See section 7.4.3.1 Parameters
	INVALID_ID Wrong appID (e.g. no app being registered with this appID value)		Code: 13	
	INVALID_DATA The data sent is invalid (out of bounds parameters, wrong JSON structure)		code: 11	
	OUT_OF_MEMORY		code: 17	The check is actually performed by SDL. Optionally, HMI may also perform the check. In this case the code must be returned
	GENERIC_ERROR The unknown issue occurred or other codes are not applicable.		code: 22	

7.4.3.1 Parameters

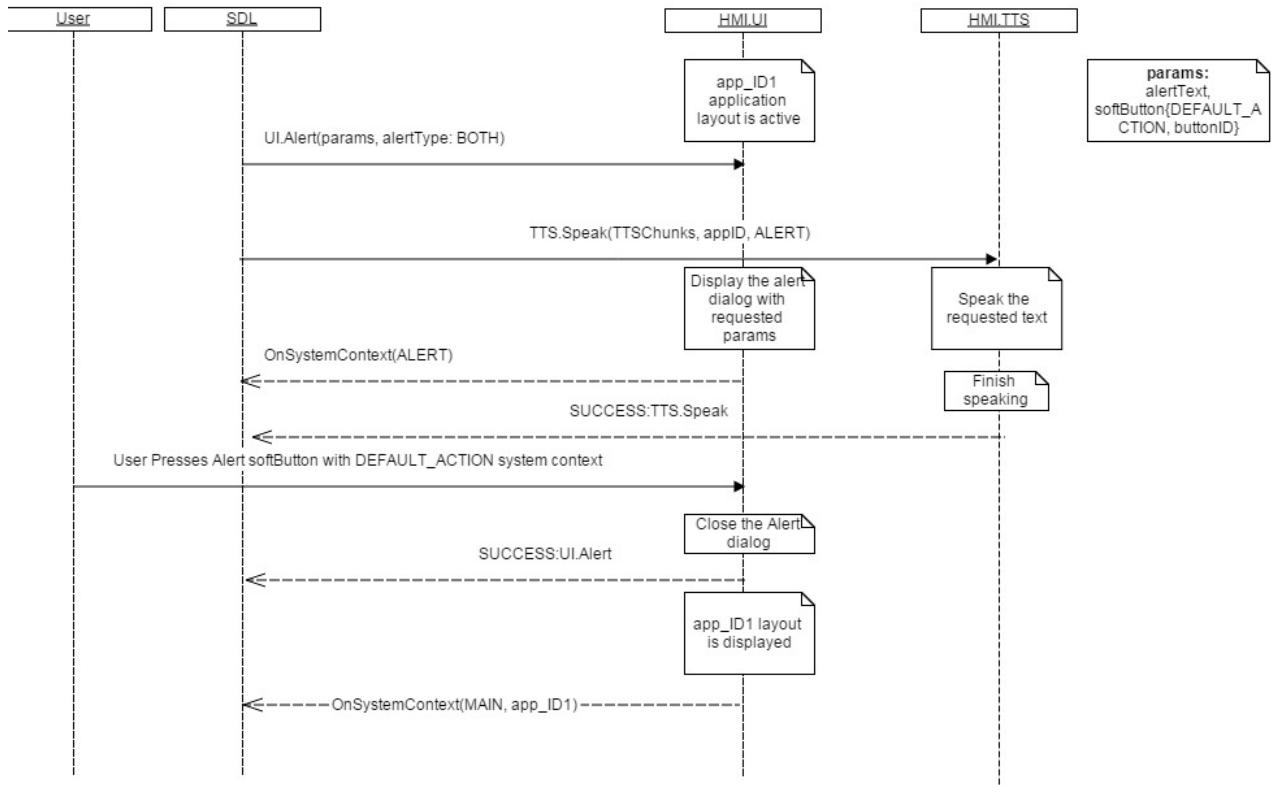
Param Name	Type	Mandatory	Additional	Description
tryAgainTime	Integer	false	Minvalue = 0 Maxvalue = 360000	Amount of time (in milliseconds) that SDL must wait before resending an alert. Must be returned <u>only</u> with REJECTED (code: 4) result.

7.4.4 Sequence Diagrams

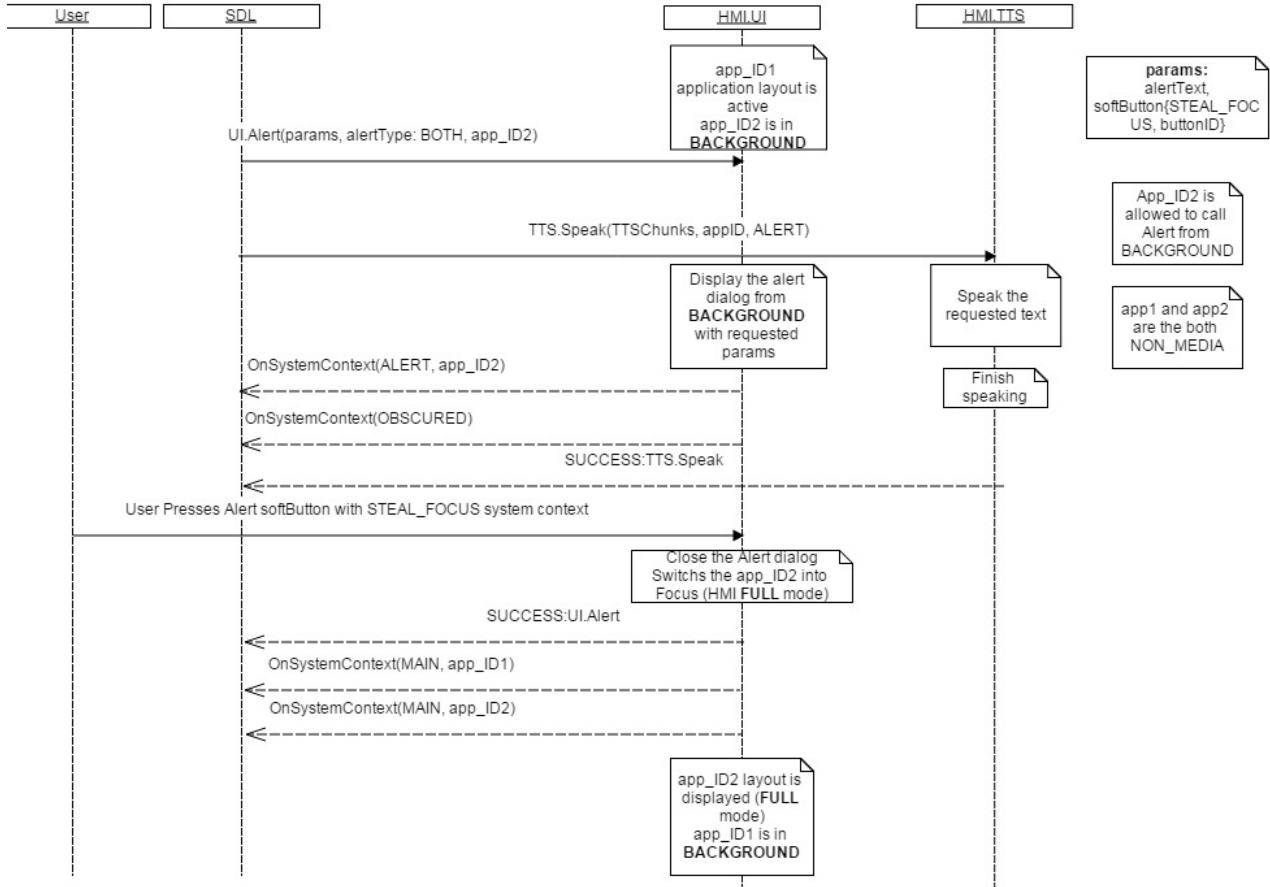
7.4.4.1 Alert together with PlayTone and Speak RPCs and closed by the timeout



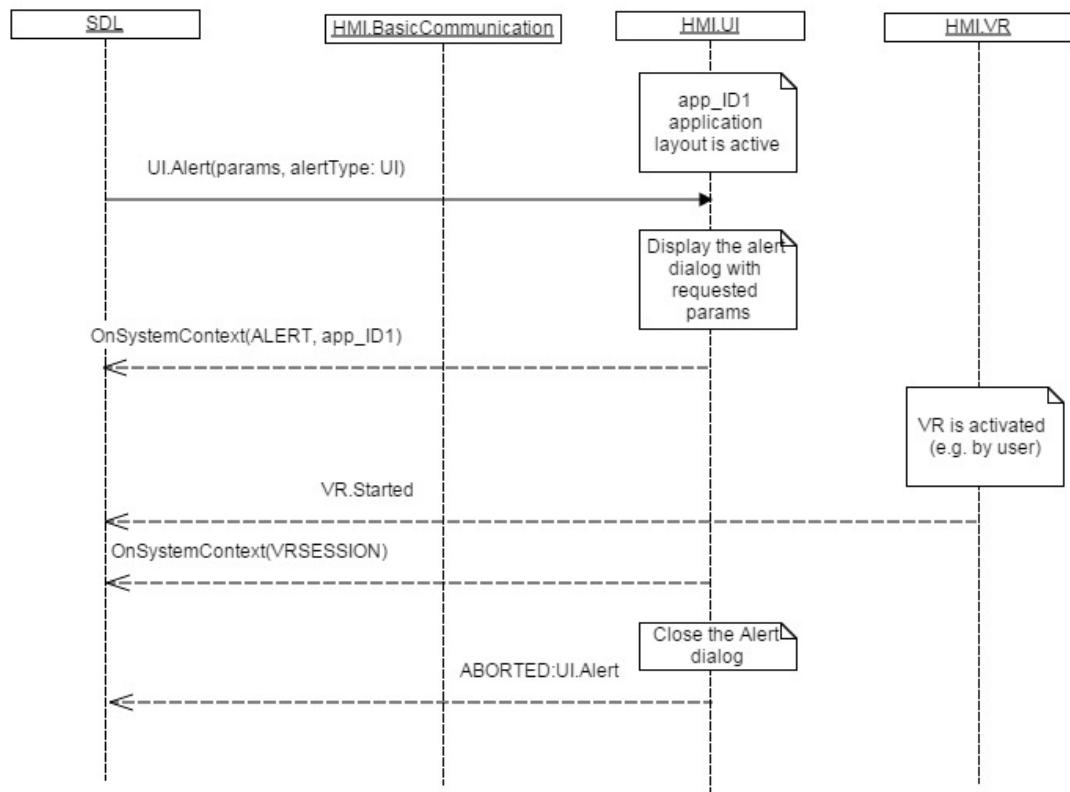
7.4.4.2 Alert displayed and closed by DEFAULT_ACTION soft button press



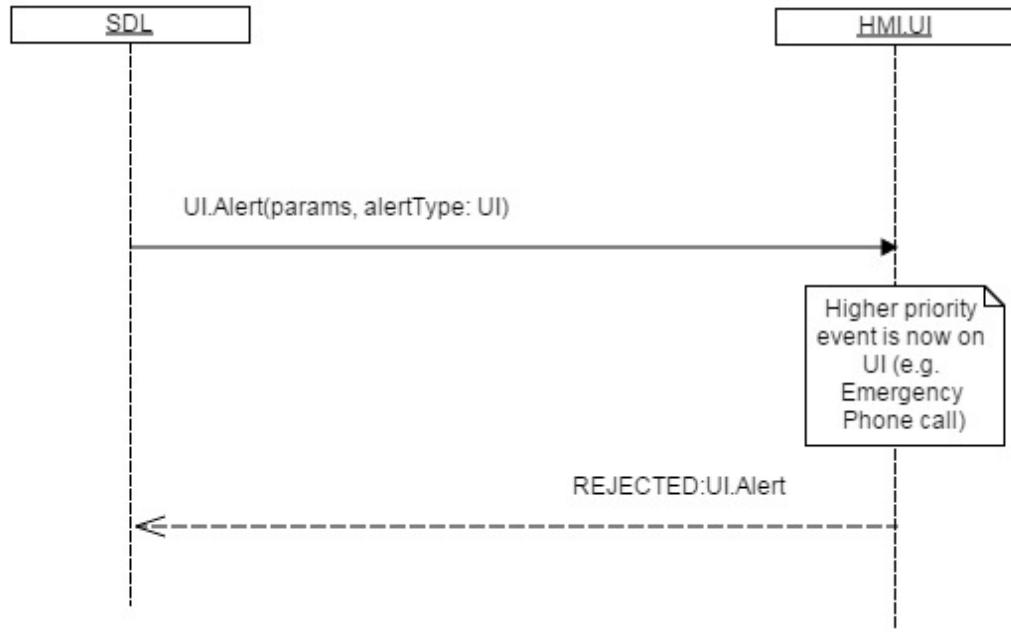
**7.4.4.3 Alert for non-active application
displayed and closed by STEAL_FOCUS soft
button press**



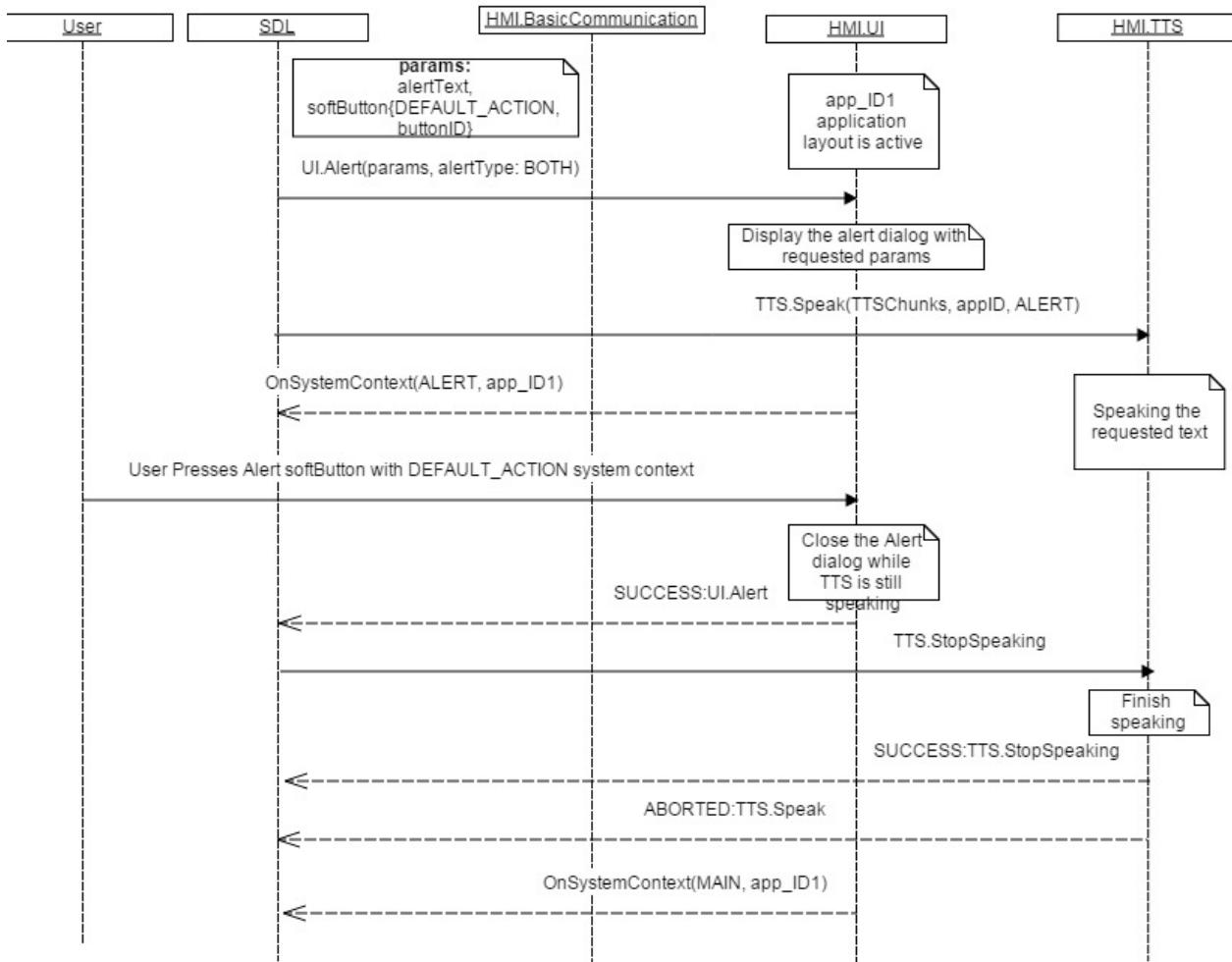
7.4.4.4 Alert is displayed and then aborted by VR session started



7.4.4.5 Alert is rejected because of RPC of a higher priority currently presented on UI



7.4.4.6 Alert BOTH when UI is closed earlier than TTS has finished speaking

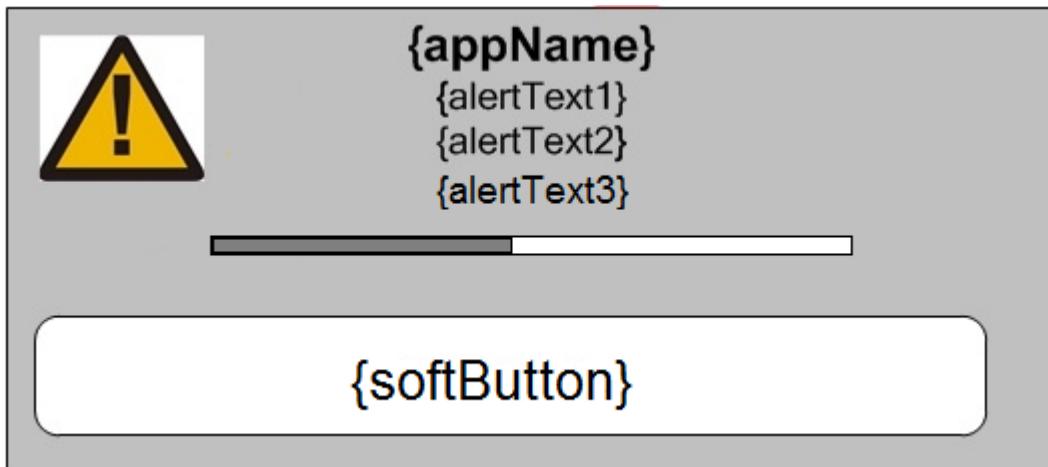


7.4.5 Possible Layout

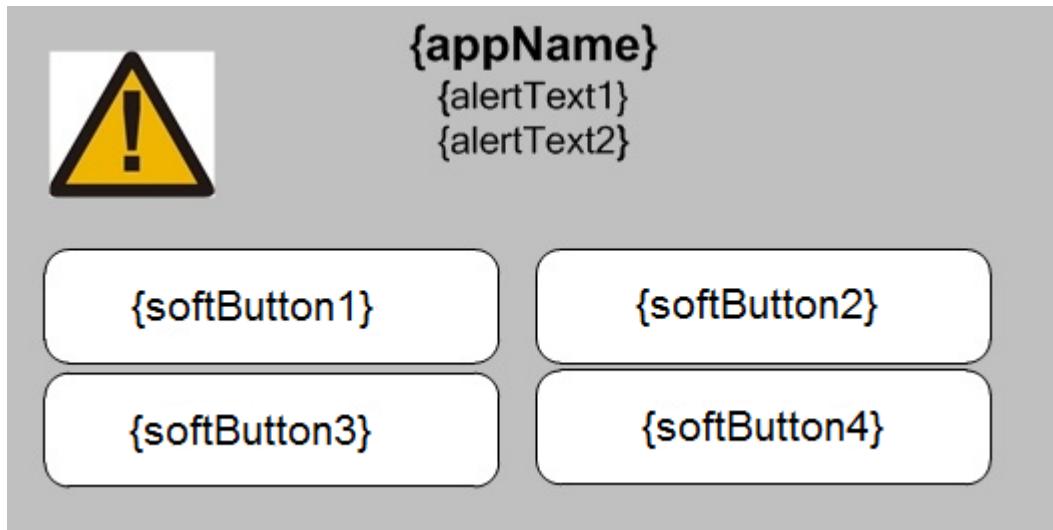
7.4.5.1 Alert dialog with text fields and NO soft buttons



7.4.5.2 Alert dialog with text fields, one soft button and a progress indicator



7.4.5.3 Alert dialog with text fields and four soft buttons



7.4.6 JSON Messages Examples

7.4.6.1 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
}
}

```

7.4.6.2 Response

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

7.4.6.3 Error message

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

7.5 Show

7.5.1 Description

Type:	Function
Sender:	SDL
Purpose:	Update the application persistent display on HMI.

With this RPC SDL requests HMI to update the supported fields for the application persistent display.

Note:

SDL uses the information received via response to [UI.GetCapabilities](#) for forming the request (uses the supported text fields names, provides the images of supported type, etc.).

The request may follow

- After the application is first activated on HMI
- For the application that is not currently active on HMI

7.5.2 Request

7.5.2.1 Behavior

HMI must:

1. Store the data sent within RPC
2. Be able to correlate the stored information with the application identified with appID.
3. Display the requested data (text, image, soft buttons, presets):
 - Right after the Show RPC arrived if the application is active on HMI
 - After the application is activated if the Show RPC:
 - Arrives while the application is not active on HMI.
 - Arrived for the active application that has then been deactivated.

Note:

Together with items requested with Show, HMI must also display the in-application commands menu (e.g. named 'Options') providing the possibility for the User to enter this menu (empty by default) and to choose among its elements (once added via [AddCommand](#) and/or [AddSubMenu](#)).

Note:

Pressing the soft button of any system action must cause
- NO changes for application related persistent display on HMI
- Sending OnButtonPress/OnButtonEvent notifications.

Note:

The sequence diagrams describing the expected HMI behavior are provided in the section 7.5.4 Sequence Diagrams

7.5.2.2 Parameters

Param Name	Type	Mandatory	Additional	Description
showString s	Common.TextFieldStruct	true	Array = true Minsize = 0 Maxsize = 7	Array of text lines to be displayed in the named text fields. The previously displayed text must not be

Param Name	Type	Mandatory	Additional	Description
				<p>changed on this RPC receipt if:</p> <ul style="list-style-type: none"> - The corresponding textfield parameter is not provided within a request. - The empty array is sent. <p>The previously displayed text must be cleared if the corresponding textfield parameter has the value of "" (empty string).</p> <p>See TextFieldStruct, TextFieldName</p>
alignment	Common.TextAlignment	false	-	<p>Specifies the way mainField1 and mainField2 texts must be aligned on the display.</p> <p>If omitted, texts must be centered.</p> <p>See TextAlignement.</p>
graphic	Common.Image	false	-	<p>Path to the optional dynamic image or the static binary image itself.</p> <p>If omitted, the displayed graphic must not be changed.</p> <p>See Image.</p>
secondaryGraphic	Common.Image	false	-	<p>Image struct determining whether static or dynamic secondary image to display in app.</p> <p>If omitted on supported displays, the displayed secondary graphic shall not change.</p>
softButtons	Common.SoftButton	false	Array = true Minsize = 0 Maxsize = 8	<p>App defined soft buttons.</p> <p>Pressing the soft button with defined system action must cause no changes on HMI.</p> <p>For all of the soft buttons the notifications OnButtonPress/OnButtonEvent must be provided by HMI.</p> <p>If omitted or the empty array is sent, the currently displayed soft button values must not be changed.</p> <p>See SoftButton.</p>
customPresets	String	false	Array = true Minsize = 0 Maxsize = 10 Maxlength = 500	<p>App labeled on-screen presets.</p> <p>If omitted or the empty array is sent, the presets must be shown as not defined (with default names "PRESET_1", "PRESET_2", "PRESET_3" etc).</p>
appID	Integer	true	-	ID of the application requested this RPC.

7.5.2.3 TextFieldStruct Structure

Param Name	Type	Mandatory	Additional	Description
fieldName	Common.TextFiledName	true	-	The name of the field where the text must be displayed in.
fieldText	String	true	Maxlength = 500	The text to be displayed.

7.5.2.4 TextFieldName Enumeration

Only the text fields applicable to Alert RPC are described within this section. All the text fields names recognized by SDL are described in the section 13.1.14 *TextFieldName*.

Element name	Short Description
mainField1	The text that must be displayed in a single or upper display line. If this value is not set, the text of mainField1 must stay unchanged. If this text is empty "", the field must be cleared..
mainField2	The text that must be displayed on the second display line. If this text is not set, the text of mainField2 must stay unchanged. If this text is empty "", the field must be cleared..
mainField3	The text that must be displayed on the second "page" first display line. If this text is not set, the text of mainField3 must stay unchanged. If this text is empty "", the field must be cleared..
mainField4	The text that must be displayed on the second "page" second display line. If this text is not set, the text of mainField4 must stay unchanged. If this text is empty "", the field must be cleared..
statusBar	The text is placed in the status bar area. Note: This relates to navigation displays If this parameter is omitted, the status bar text must remain unchanged. If this parameter is an empty string, the field must be cleared. If provided and the display has no status bar, this parameter must be ignored..
mediaClock	Text value for MediaClock field. Shall arrive in the form as described in the MediaClockFormat enumeration If this text is set, any automatic media clock updates previously set with SetMediaClockTimer must be stopped..
mediaTrack	The text that should be displayed in the track field. This field should be valid only for media applications on. If this text is not set, the text of

Element name	Short Description
	mediaTrack must stay unchanged. If this text is empty "", the field must be cleared. ..

7.5.2.5 TextAlign Enumeration

Element name	Short Description
LEFT_ALIGNED	Text of mainField1 and mainField2 is left aligned.
RIGHT_ALIGNED	Text of mainField1 and mainField2 is right aligned.
CENTERED	Text of mainField1 and mainField2 is centered.

7.5.3 Response

Note:

There is a difference in message type for WebSocket and D-Bus connection (see the table below).

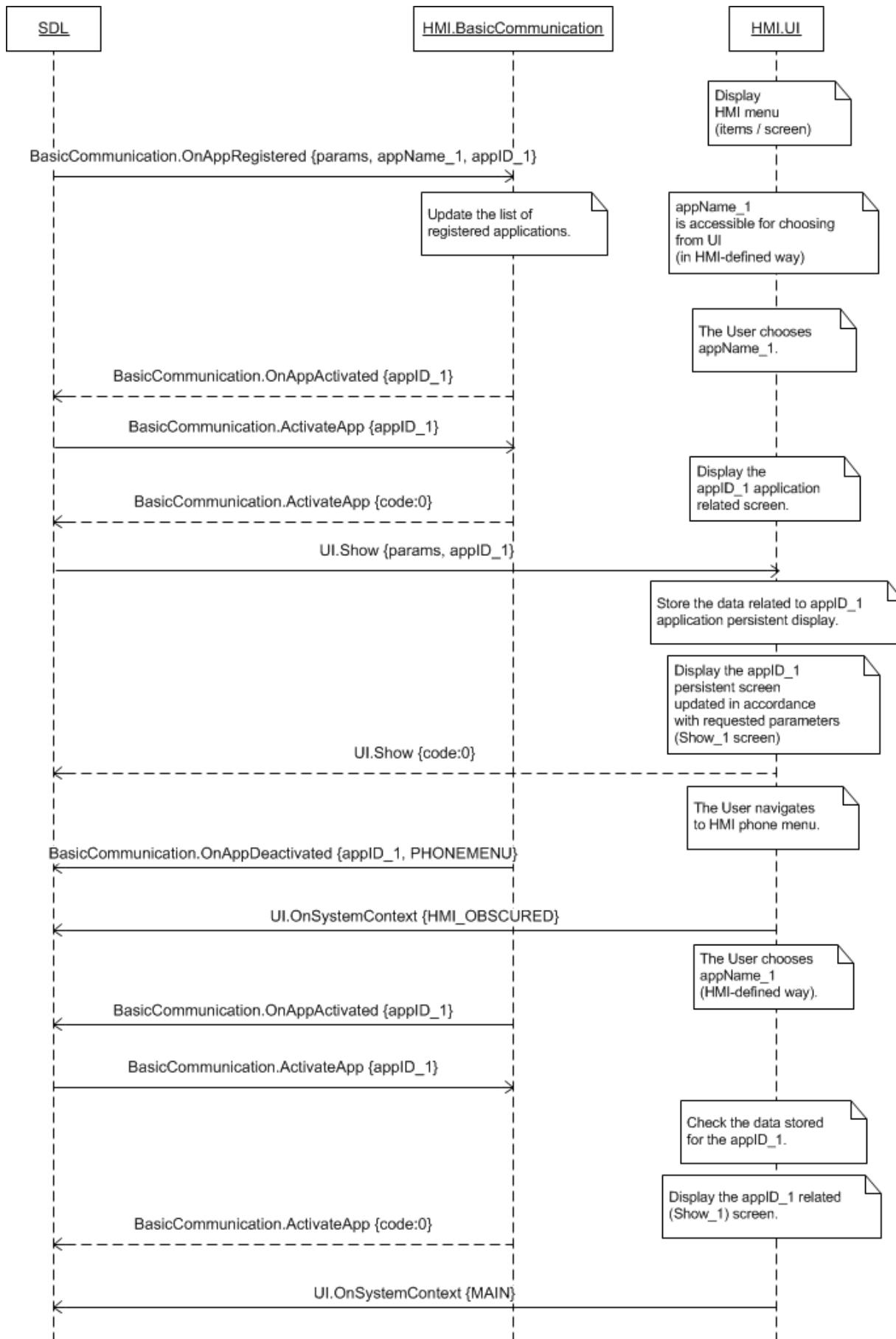
Result	Description	Message type		Message Params	Notes
		WebSocket	D-Bus		
Success	SUCCESS HMI has stored the requested data correlating it with the application ID and updated the display areas if the requested application is currently active.	JSON response	Method return	code: 0	
	UNSUPPORTED_RESOURCE When images are sent by SDL but they aren't supported by HMI or HMI doesn't support the type of images sent (STATIC/DYNAMIC). The request is considered to be successfully executed anyway if the the request informed the user with other Show information.			code: 2	

Failure	REJECTED HMI is expected to return REJECTED result code in case HMI is currently busy with a higher-priority event.	JSON error message	Method return	code:4	
	INVALID_ID Wrong appID (e.g. no app being registered with this appID value)			code:13	
	INVALID_DATA The data sent is invalid (invalid JSON syntax, parameters out of bounds or of wrong type)			code:11	Applicable for this RPC result codes. Please see Result Enumeration for all SDL-supported codes.
	GENERIC_ERROR : 1) The unknown issue occurred or other codes are not applicable.			code:22	

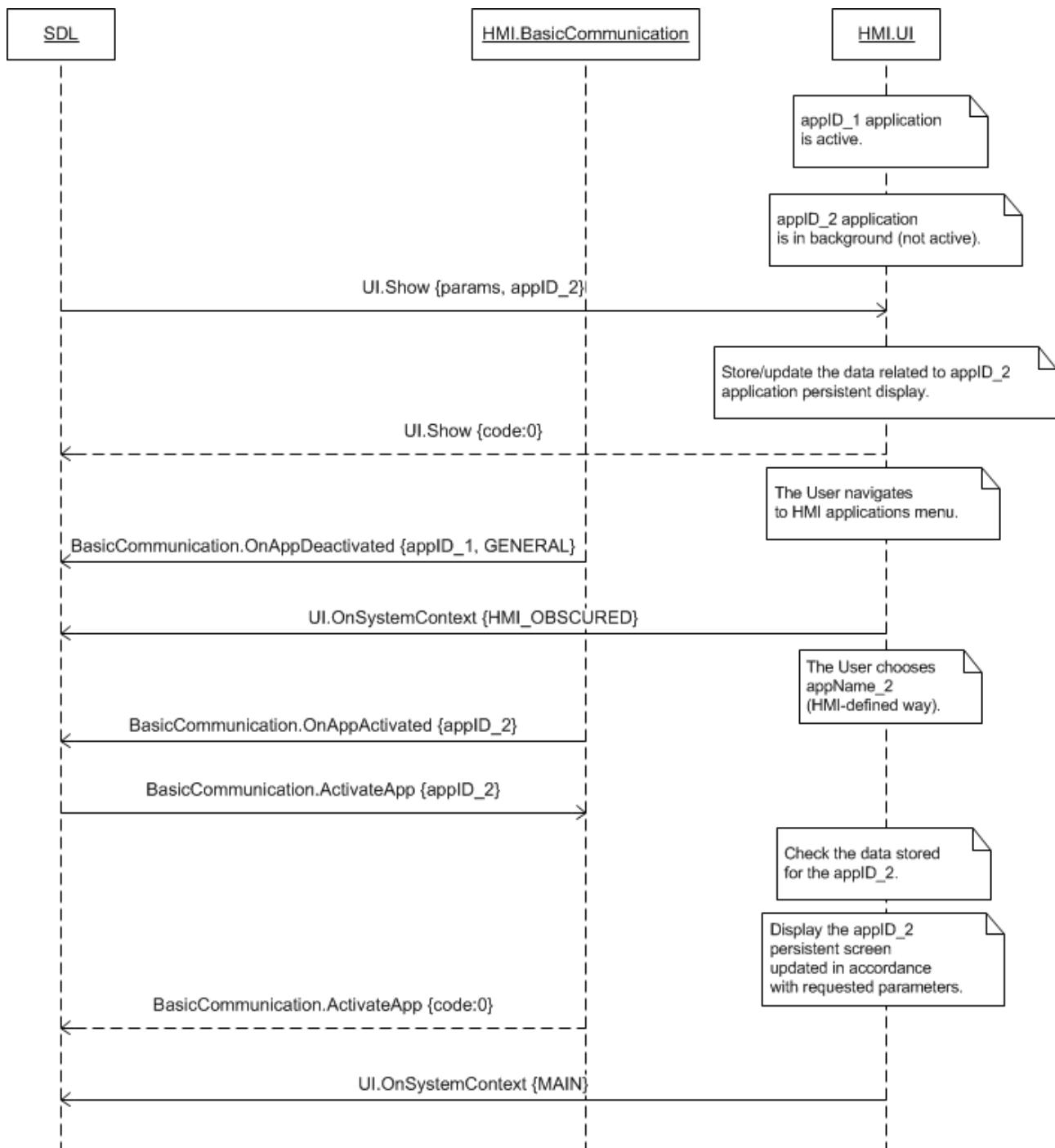
SDL Note: In case HMI does not respond SDL's request during SDL-default timeout (10 sec), SDL will return GENERIC_ERROR result code to the corresponding mobile app's request.

7.5.4 Sequence Diagrams

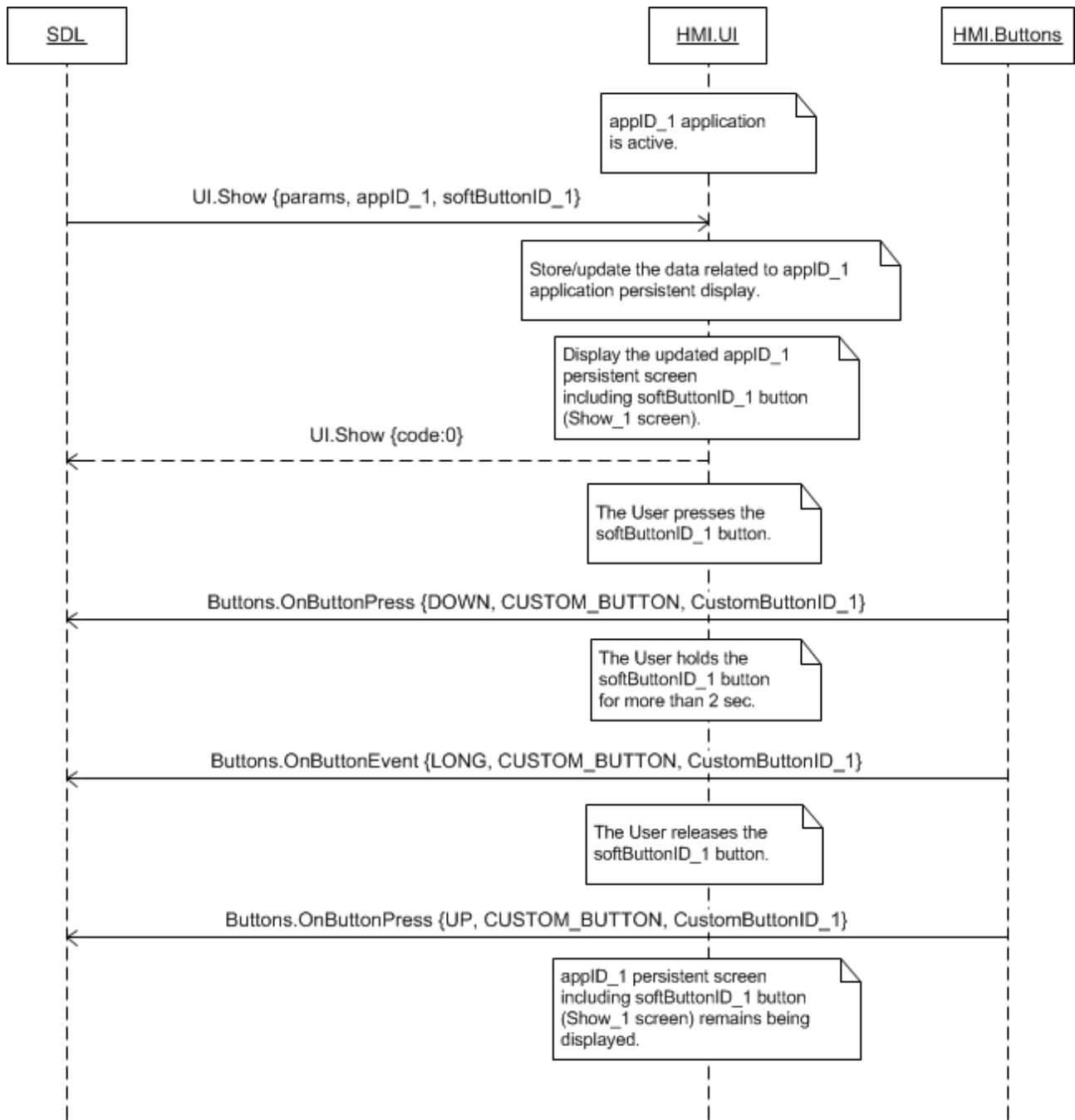
7.5.4.1 Show for the active application that is then deactivated and activated again



7.5.4.2 Show for the application currently not active on HMI



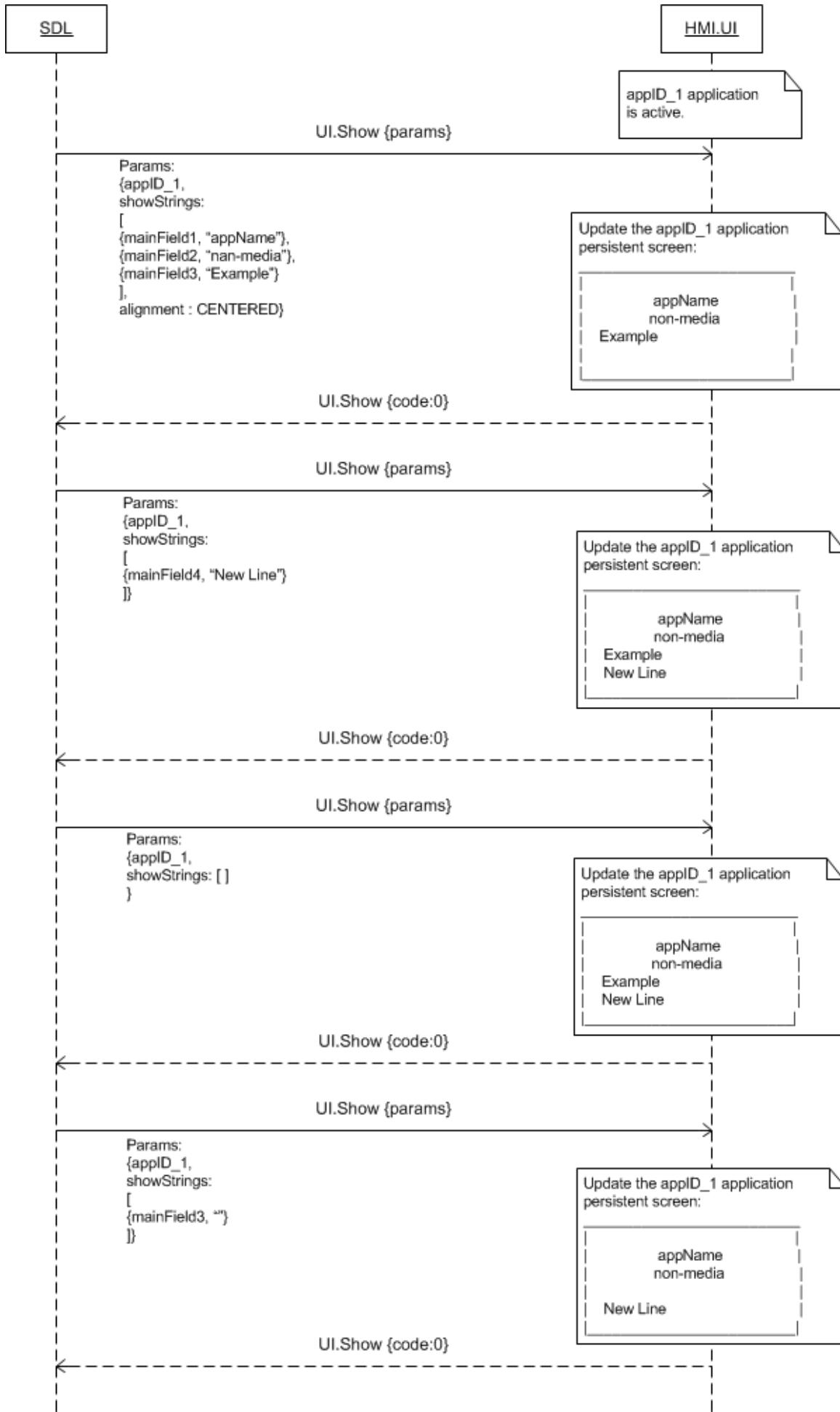
7.5.4.3 Requested with Show soft button press



Note:

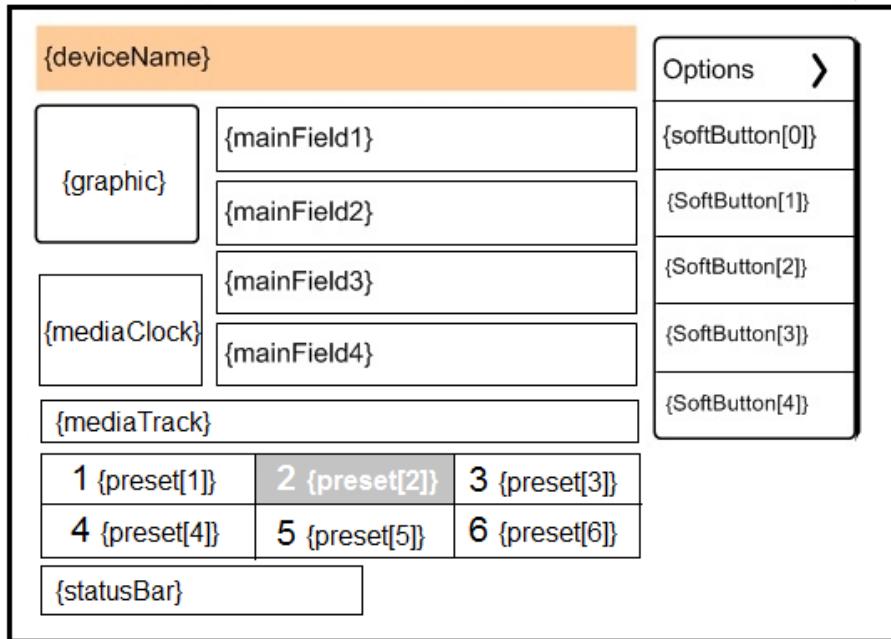
*CustomButtonID parameter returned via
OnButtonEvent/OnButtonPress notifications must
have the same value as the softButtonID parameter
sent via Show RPC.*

7.5.4.4 Show text fields expected behavior

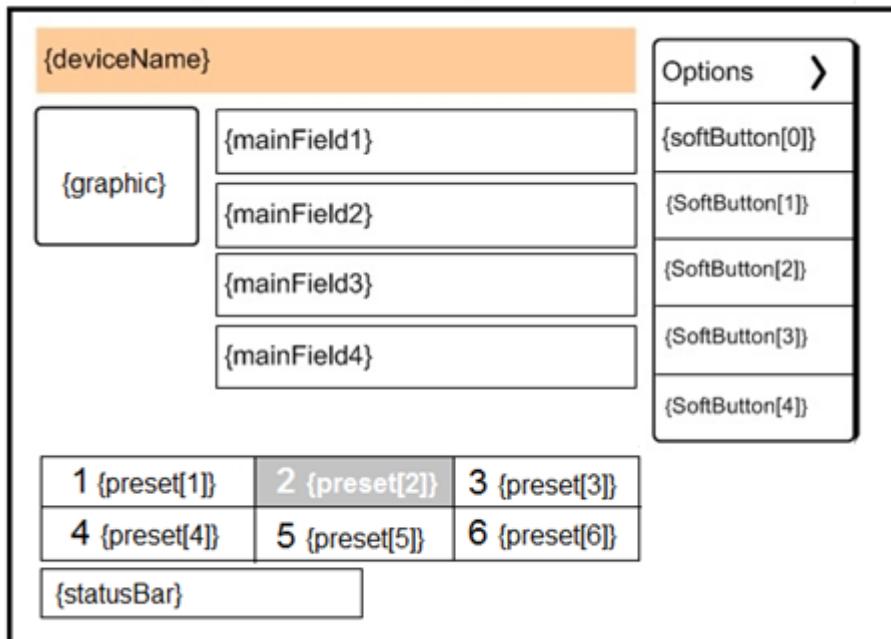


7.5.5 Possible Layout

7.5.5.1 Persistent display areas updated with Show for media applications



7.5.5.2 Persistent display areas updated with Show for non-media applications



7.5.6 JSON Messages Examples

7.5.6.1 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
Song"]
    }
}
```

```

        Album"] ,
        "appID" : 8726
    }
}

```

7.5.6.2 Response

```

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

```

7.5.6.3 Error message

```

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

```

7.6 AddCommand

7.6.1 Description

Type:	Function
Sender:	SDL
Purpose:	Add a command to the in-application menu or sub menu.

This RPC represents a request to HMI to add a command to in-application menu or sub-menu (the latter follows after AddSubMenu RPC).

The request may arrive in both cases of activated and deactivated application on HMI.

7.6.2 Request

7.6.2.1 Behavior

HMI must:

1. Provide the in-application menu (e.g. named 'Options') displayed right after the application is activated on HMI.

2. Provide the possibility for the User to enter this menu (empty by default) and to choose among its elements (once added via `UI.AddCommand` and/or `AddSubMenu`).

3. Check that the limit of menu / sub menu (depending on where the command is requested to be added to, see point 5. below) items is not exhausted by the command being requested (and if so, reject the request with the corresponding result code).

4. Store the data provided within this RPC correlating it with the `appID`.

5. Add the command with requested parameters (name, position) to:

- The top level menu (e.g. ‘Options’) if `parentID` is not provided within RPC.
- The sub menu with the ID corresponding to the `parentID` provided within RPC.

6. Provide the response correspondingly to the result of RPC execution.

Note:

The applicable to this RPC result codes are provided in section [7.8.3 Response](#).

7. Provide the `UI.OnCommand` notification with the `cmdID` corresponding to the command chosen by the User.

Important Note:

Once commands are added for the named application, they must remain accessible for the User (until `UI.DeleteCommand` comes) when this application is activated after having been deactivated.

In case the app sends `AddCommand` with the both `UI` and `VR` portions and adding one of the portions ended with erroneous response or no response at all, `SDL` must send `DeleteCommand` for a successfully added portion(`VR` or `UI`) in the following cases (see diagrams for more details [7.6.4.6-7.6.4.9](#) :

3. *In case `SDL` sends both `UI.AddCommand` and `VR.AddCommand` with one and the same `cmdID` to `HMI` AND **`UI.AddCommand gets successful response`** from `HMI` in return AND `VR.AddCommand` gets any *erroneous response*/no response from `HMI` - `SDL` must send **`UI.DeleteCommand`** for the successfully added `cmdID` to `HMI`.*
4. *In case `SDL` sends both `UI.AddCommand` and `VR.AddCommand` with one and the same `cmdID` to `HMI` AND **`VR.AddCommand gets successful response`** from `HMI` in return AND `UI.AddCommand` gets *erroneous response except of `WARNINGS` and `UNSUPPORTED_RESOURCE`*/ no response from `HMI` - `SDL` must send **`VR.DeleteCommand`** for the successfully added `cmdID` to `HMI`*

7.6.2.2 Parameters

Param Name	Type	Mandatory	Additional	Description
cmdID	Integer	true	Minvalue = 0	<p>ID of the command to be added.</p> <p>This ID is unique within the application related to this request.</p> <p>This ID:</p> <ul style="list-style-type: none"> - Must be returned by HMI later within <code>OnCommand</code> notification to identify the command selected by the user. - Might be provided by SDL later within <code>DeleteCommand</code> request.
menuParams	Common.MenuParams	false	-	<p>Defines the name of the command, its position, ID of the submenu for the command to be added to.</p> <p>If not provided, the command must be added to the end of the list of items of the top-level menu.</p> <p>See <code>MenuParams</code>.</p>
cmdIcon	Common.Image	false	-	<p>Image to be displayed for representing the command: either the path to dynamic image stored on HU or the static binary image itself.</p> <p>If not provided, no image or the default one if applicable should be displayed.</p> <p>See <code>Image</code>.</p>
appID	Integer	true	-	ID of the application related to this RPC.

7.6.2.3 MenuParams

Param Name	Type	Mandatory	Additional	Description
parentID	Integer	false	minvalue = 0 maxvalue = 2000000000	<ul style="list-style-type: none"> - Unique ID of the sub menu, the command must be added to. - If not provided, the command must be added to the top-level application menu.
position	Integer	false	minvalue = 0 maxvalue = 1000	<p>This value is the position within the elements of top-level application menu / sub menu where the command must be added to:</p> <ul style="list-style-type: none"> - If 0, the item must be inserted to the first position. - If 1, the item must be inserted to the second position. - Etc. <p>If the next command comes with the same position value, it must be added as the next item within corresponding 'position block'. (e.g. the command with position 0 must be added</p> <ul style="list-style-type: none"> - after the previous item with position 0 - and before the first item with position 1). <p>If the value is greater than or equal to the number of elements of the top-level application menu, the</p>

				command must be appended to the end of the list. If omitted the item must be added to the end of the list.
menuName	String	true	maxlength = 500	The text that must be shown as a name of a command.

7.6.3 Response

Note:

There is a difference in message type for WebSocket and D-Bus connection (see the table below).

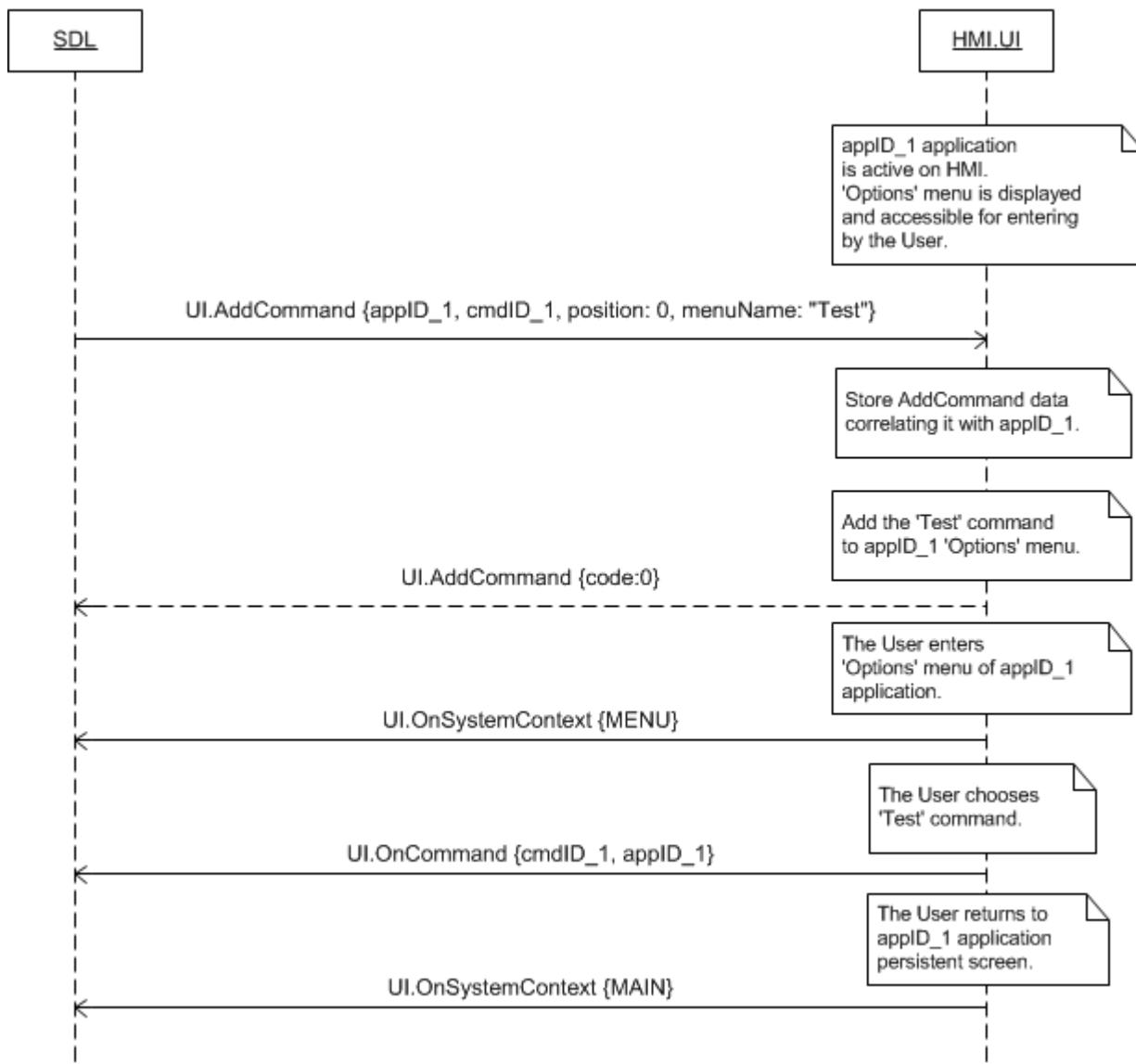
Result	Description	Message type		Message Params	Notes
		WebSocket	D-Bus		
Success	HMI has stored the data correlating it with appID and added the command with the corresponding parameters to the in-application menu/sub menu.	JSON response	Method return	code: 0	
	UNSUPPORTED_RESOURCE When images are sent by SDL but they aren't supported by HMI or HMI doesn't support the type of images sent by SDL (STATIC/DYNAMIC).			Code: 2	
Failure	REJECTED 1) The limit of position items of the top level menu is exhausted. 2) The limit of position items of the corresponding sub menu the command to be added to is exhausted.	JSON error message	Method return	Code: 4	Applicable for this RPC result codes. Please see Result Enumeration for all SDL-recognized codes.
	INVALID_DATA The data sent is invalid (invalid JSON syntax, parameters out of bounds or of wrong type)			code: 11	
	INVALID_ID 1) The command with requested cmdID is already added for the named application. 2) The sub menu with requested parentID does not exist.			code: 13	
	GENERIC_ERROR: 1) The unknown issue occurred or other codes are not applicable.			code: 22	

--	--	--	--	--	--

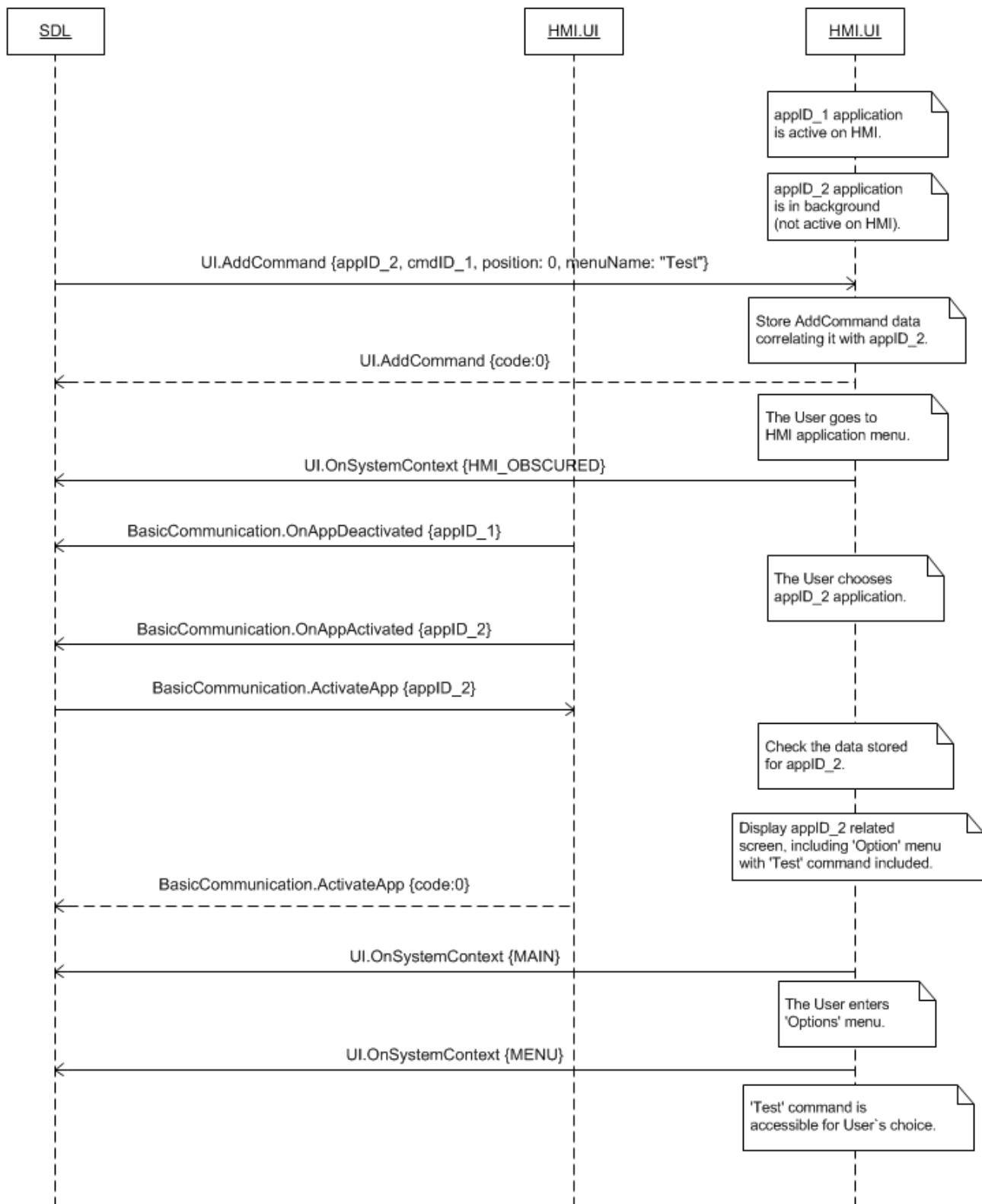
SDL Note: In case HMI does not respond SDL's request during SDL-default timeout (10 sec), SDL will return `GENERIC_ERROR` result code to the corresponding mobile app's request.

7.6.4 Sequence Diagrams

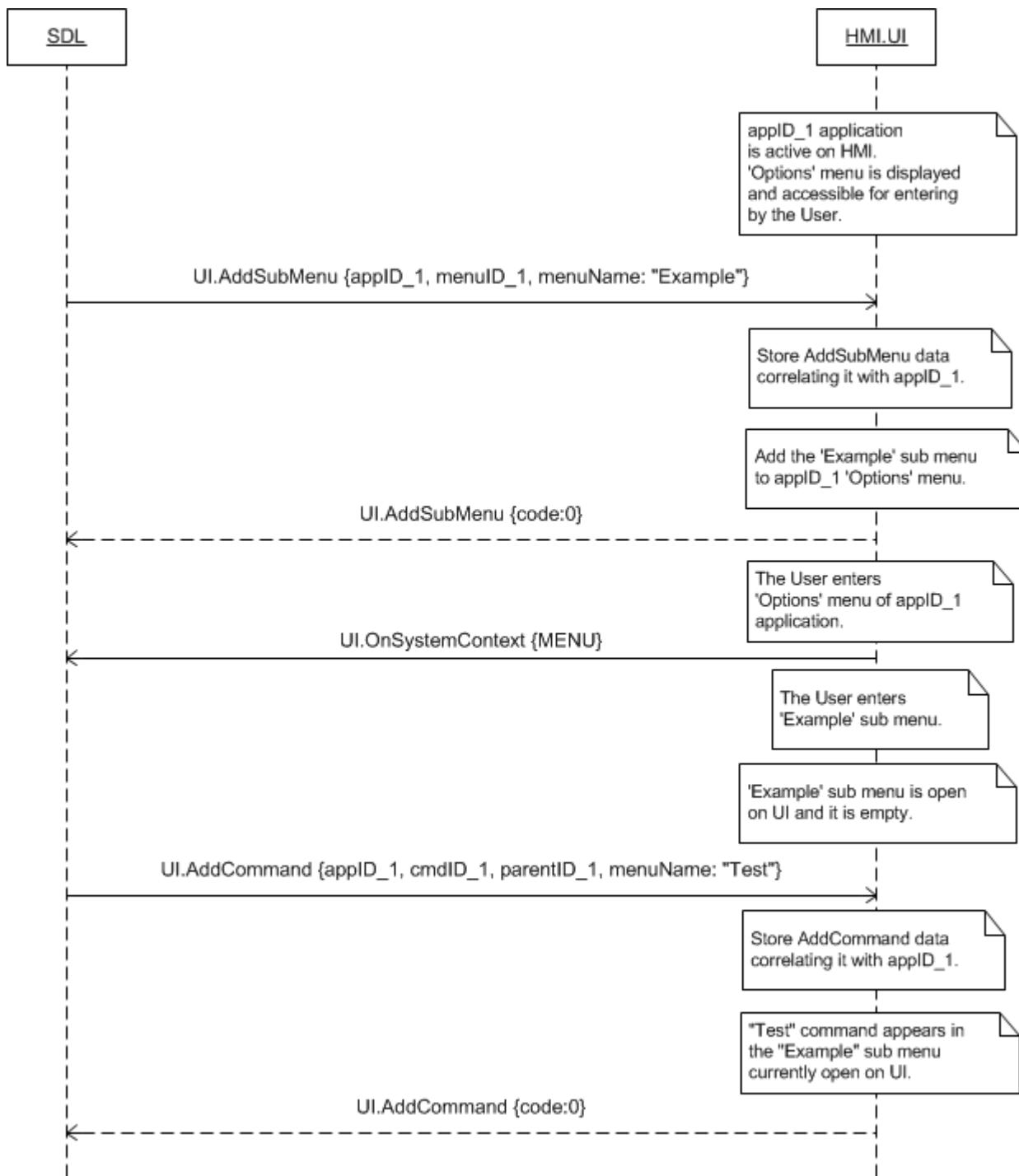
**7.6.4.1 AddCommand for the active application
on HMI, the command is chosen by the User.**



7.6.4.2 AddCommand for the application not active on HMI



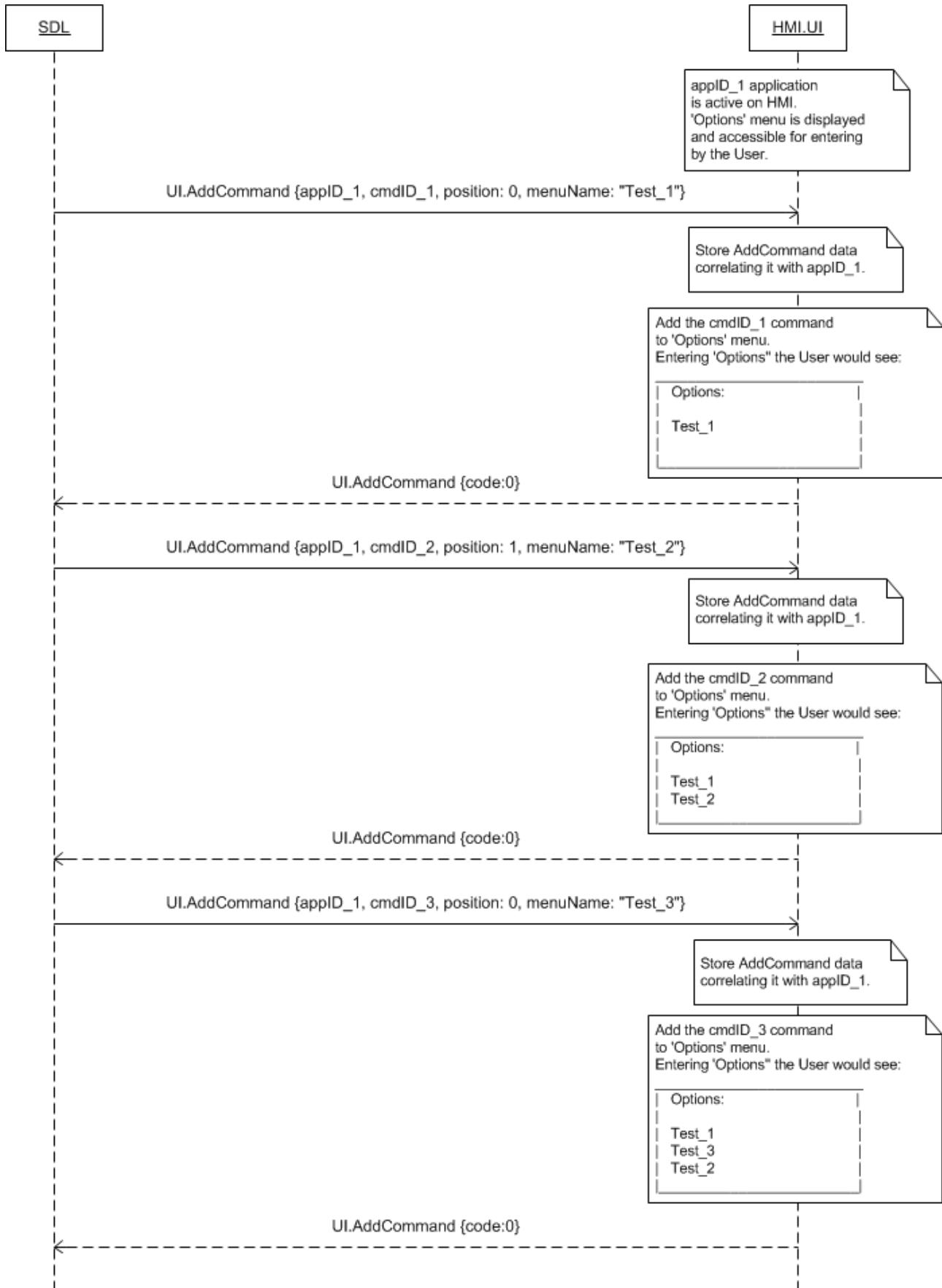
7.6.4.3 AddCommand: adding command to sub menu



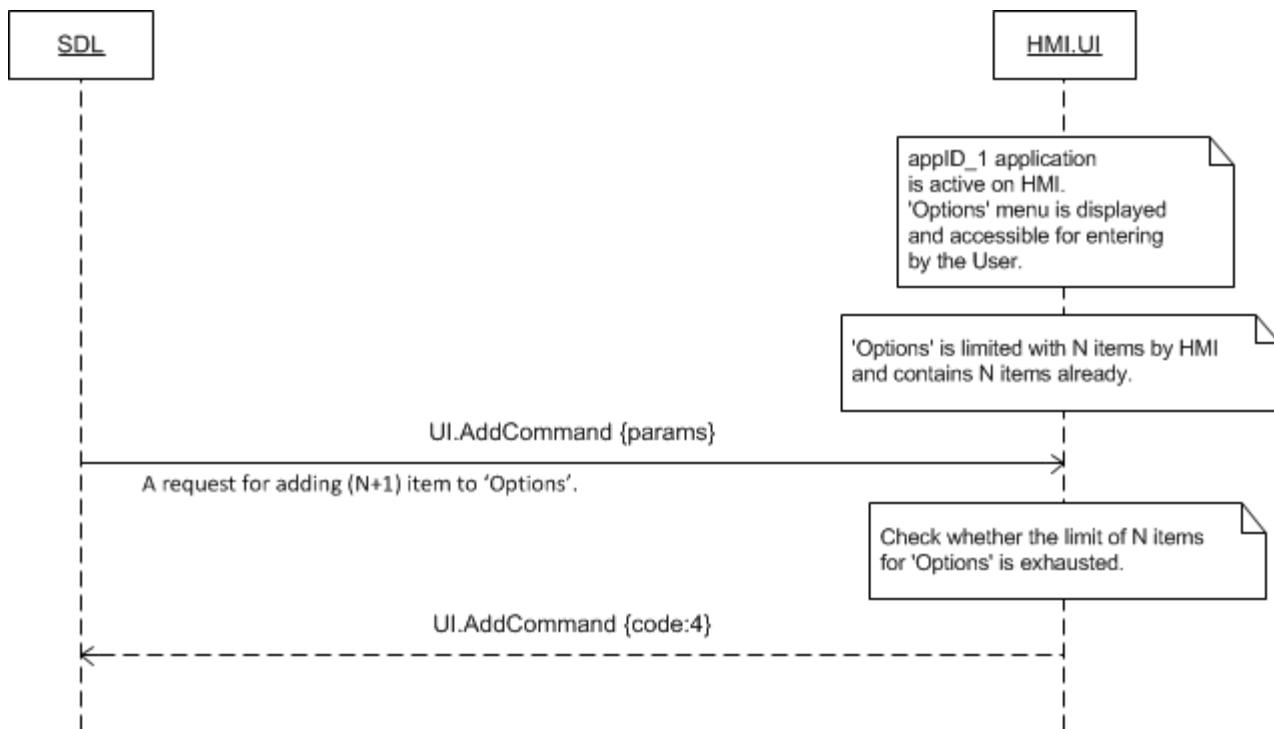
Note:

The value of `parentID` sent via `UI.AddCommand` is equal to the value of `menuID` previously provided via `AddSubMenu`.

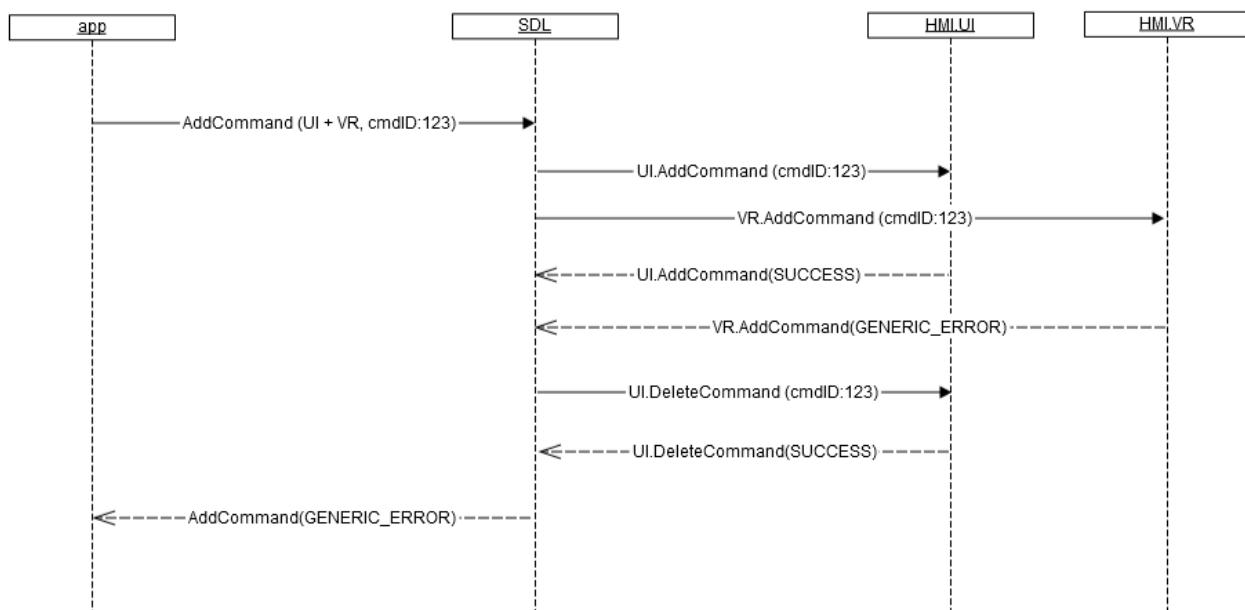
**7.6.4.4 AddCommand: expected behavior of
adding commands depending on position
parameter**



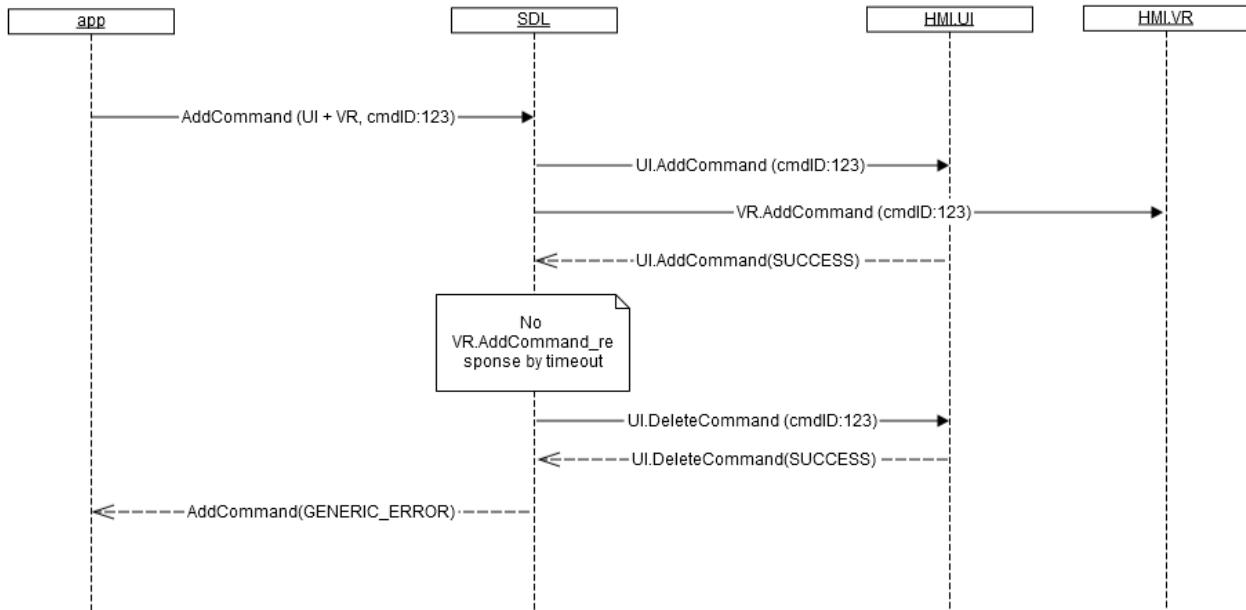
7.6.4.5 AddCommand rejected because of the limit of menu items exhausted



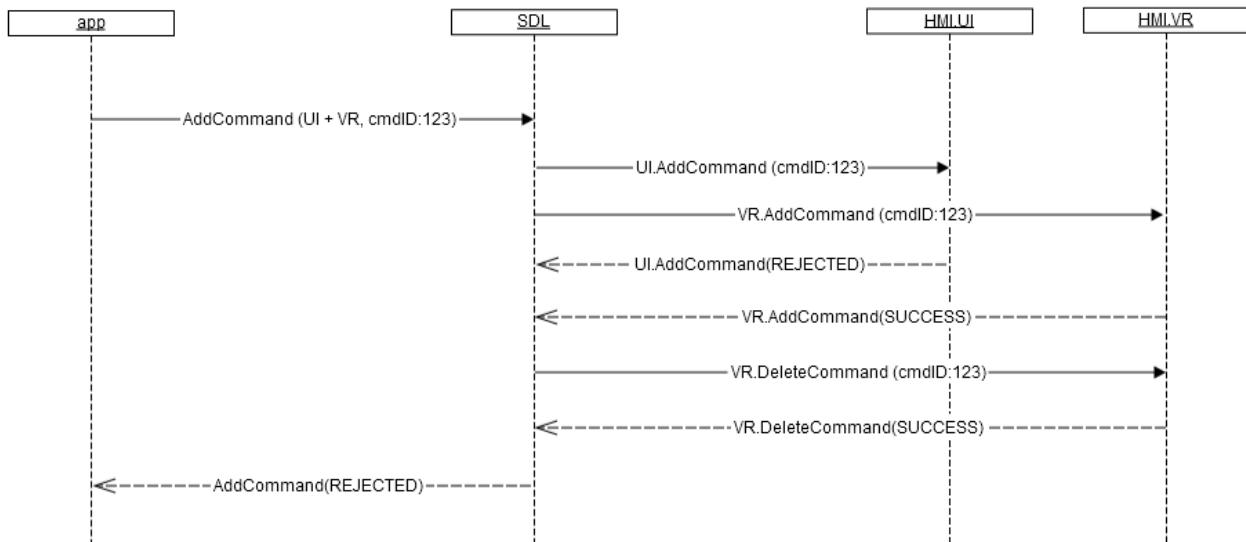
7.6.4.6 UI.AddCommand returns SUCCESS, VR portion failed



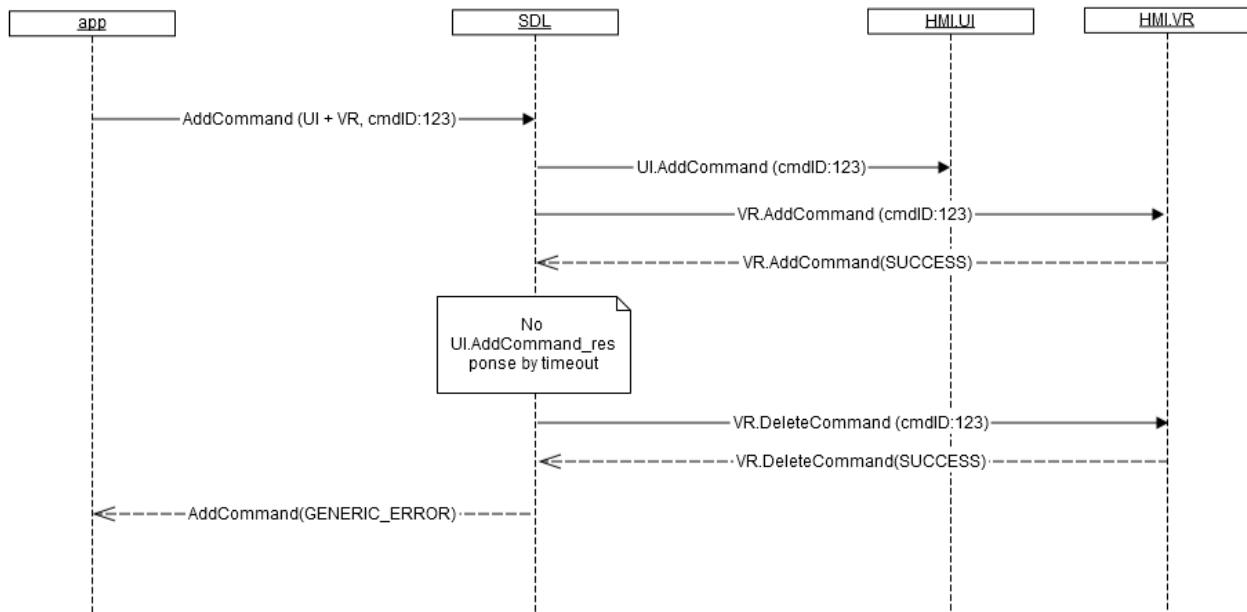
**7.6.4.7 UI.AddCommand returns SUCCESS,
VR portion no response**



**7.6.4.8 UI.AddCommand fails, VR portion
SUCCESS**

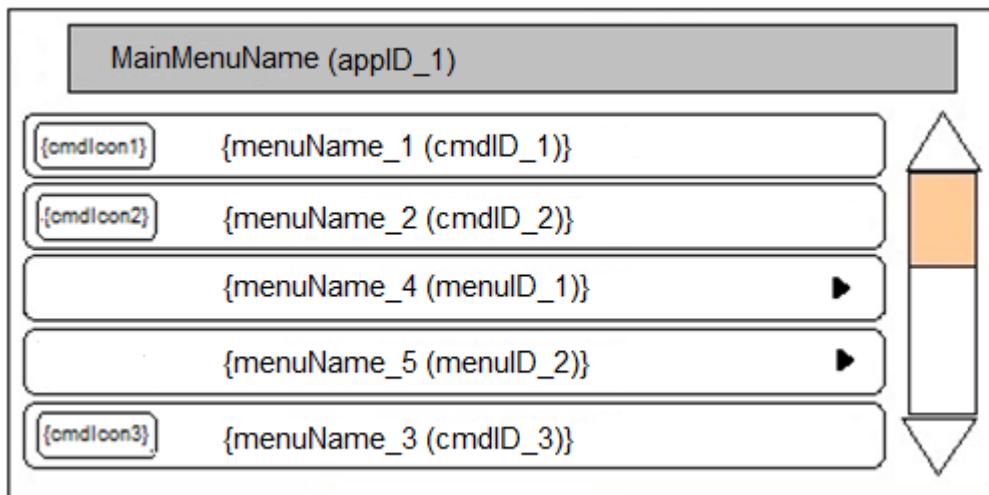


7.6.4.9 UI.AddCommand no response, VR portion SUCCESS

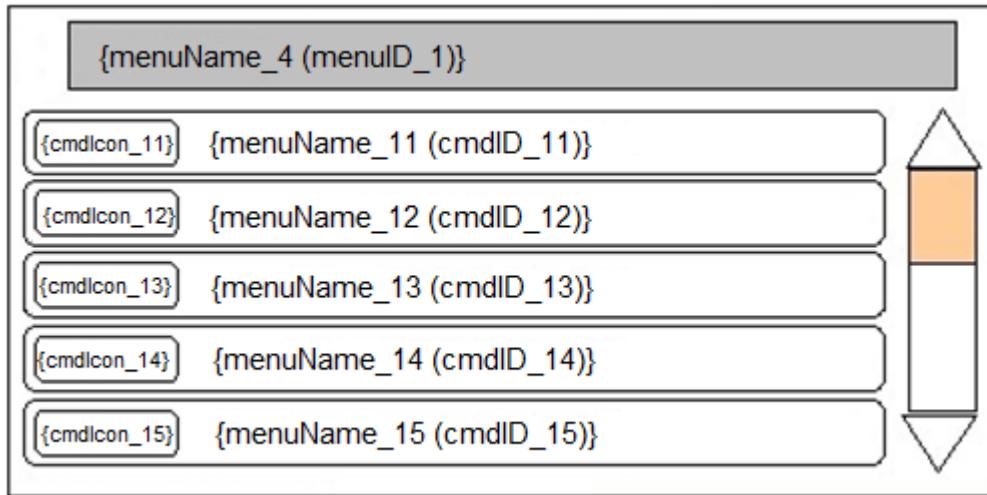


7.6.5 Possible Layout

7.6.5.1 Application main menu with sub menus and commands



7.6.5.2 Application sub menu with commands



7.6.6 JSON Messages Examples

7.8.6.1 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  
    }  
}
```

7.6.6.2 Response

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

7.6.6.3 Error message

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

7.7 DeleteCommand

7.7.1 Description

Type:	Function
Sender:	SDL
Purpose:	Delete the command from in-application menu

SDL requests to delete the command from in-application menu or sub menu of the named application previously added via [UI.AddCommand](#).

The request may be executed for the application being in focus the same as in a background on HMI. The permissions to perform DeleteCommand on a specified application HMI level (e.g. FULL or BACKGROUND) is regulated by SDL Policy Manager.

7.7.2 Request

7.8.2.1 Behavior

HMI must:

1. Update pull of the application UI commands according to the RPC parameters.
2. Delete the command identified with cmdID.
3. Display updates:
 - As soon as the named application is/has become active and the corresponding menu/sub menu is/become opened on UI (e.g. as a result of User's actions)
5. Provide the response corresponding to the result of RPC execution.

Note:

The applicable to this RPC result codes are provided in section 7.7.3 Response.

Note:

- The value of cmdID is previously sent to HMI via UI.AddCommand.

- The value of appID is previously sent to HMI via UpdateAppList or OnAppRegistered.

7.7.2.2 Parameters

Param Name	Type	Mandatory	Description
cmdID	Integer	true	ID of the command to be deleted (the one sent within the AddCommand request).
appID	Integer	true	ID of the application that concerns this RPC.

7.7.3 Response

The result codes applicable to DeleteCommand response:

Note:

There is a difference in message type for WebSocket and D-Bus connection (see the table below).

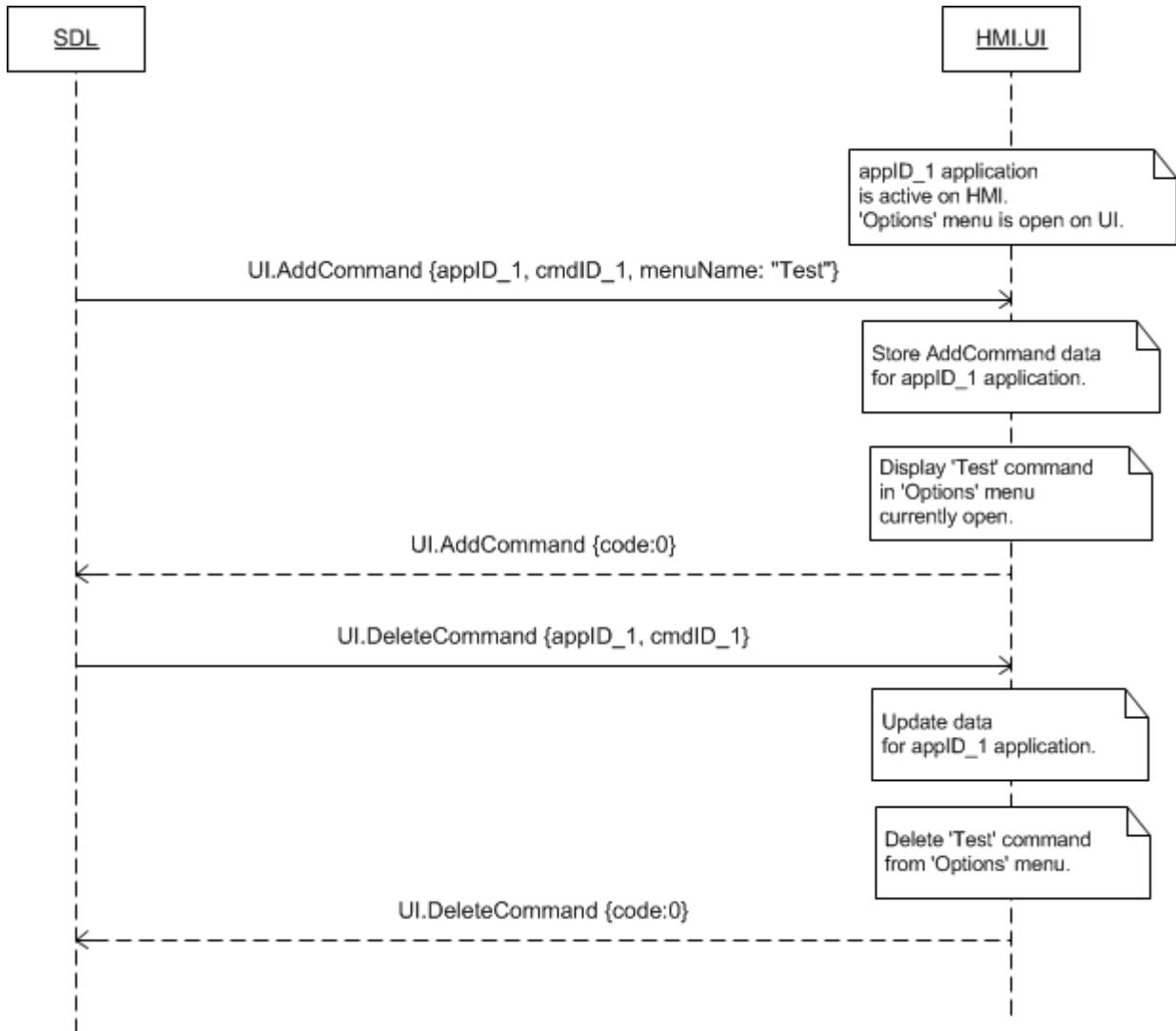
Result	Description	Message type		Message Params	Notes
		WebSocket	D-Bus		
Success	SUCCESS HMI has updated the named application related data and deleted the requested command.	JSON response	Method return	code : 0	.
Failure	INVALID_DATA The data sent is invalid (invalid JSON syntax, parameters are out of bounds or of wrong type)	JSON error message	Method return	code : 11	Applicable for this RPC result codes. Please see Result Enumeration for all SDL-recognized codes.
	INVALID_ID The command with requested cmdID does not exist on HMI for the named application. The app			Code : 13	

	with appID in the request isn't registered			
	GENERIC_E RROR: 1) The unknown issue occurred or other codes are not applicable.		code : 22	

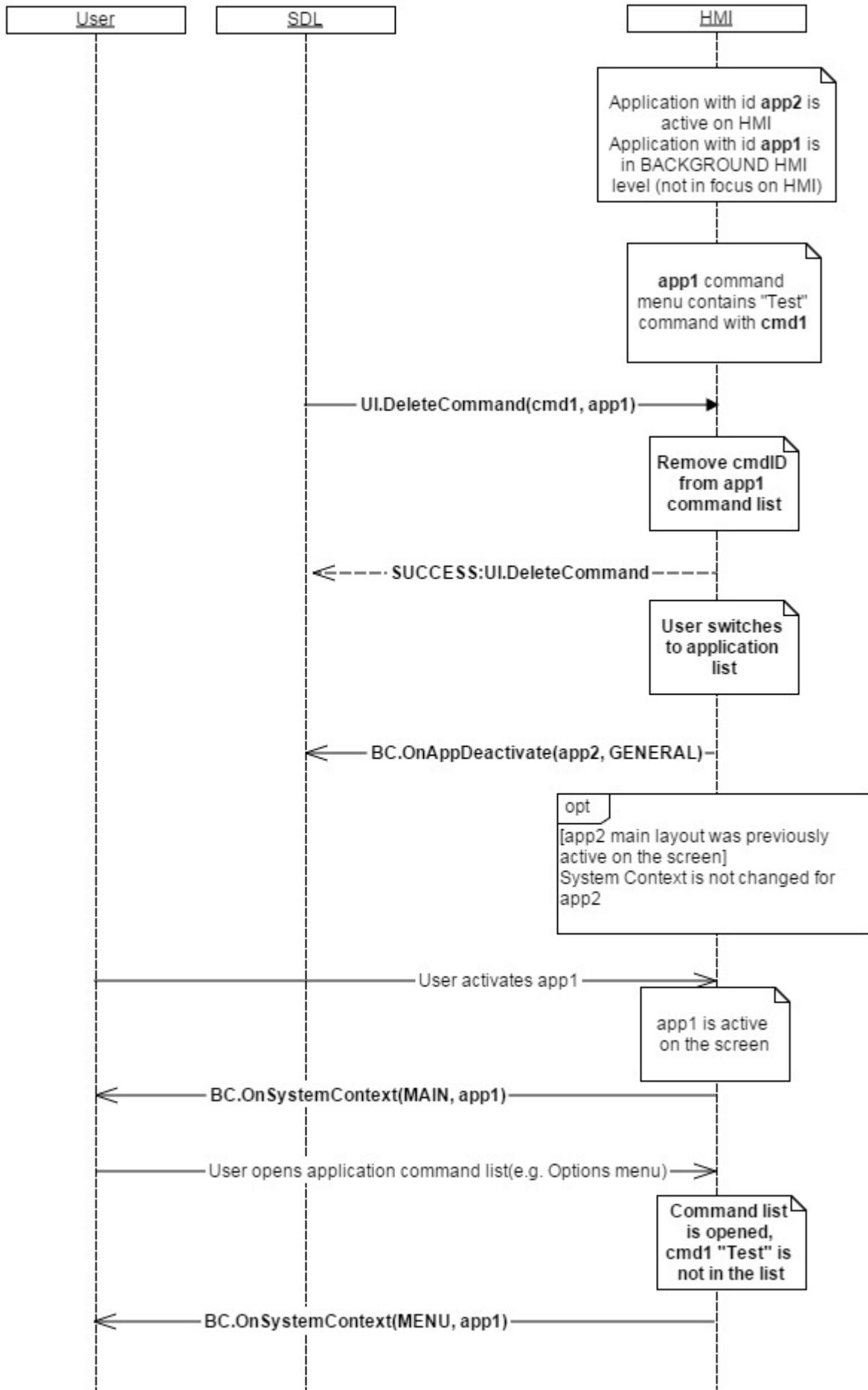
SDL Note: In case HMI does not respond SDL's request during SDL-default timeout (10 sec), SDL will return GENERIC_ERROR result code to the corresponding mobile app's request.

7.7.4 Sequence Diagrams

7.7.4.1 DeleteCommand, for the application active on HMI, for the command from the menu currently open on UI and preceding AddCommand



7.7.4.2 DeleteCommand for the application not active on HMI



7.7.5 JSON Messages Examples

7.7.5.1 Request

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

7.7.5.2 Response

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

7.7.5.3 Error message

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

7.8 AddSubMenu

7.8.1 Description

Type:	Function
Sender:	SDL
Purpose:	Add a sub menu to in-application menu

This RPC represents a request to HMI to add a command into in-application main menu. The sub menu is never requested to be added into other sub menu.

Further, SDL may request to add commands via AddCommands RPC to already created sub menu identified with menuID.

The request may arrive in both cases of activated and deactivated application on HMI (depends on Policy Table permissions, by default allowed to operate in all HMI levels except of NONE).

7.8.2 Request

7.8.2.1 Behavior

HMI must:

- 1) Provide the in-application main menu (e.g. named 'Options') available on layout right after the application is activated on HMI.
- 2) Provide the possibility for the User to enter this menu (empty by default) and to choose among its elements (once added via UI.AddCommand and/or AddSubMenu).
- 3) Check that the limit of the application menu items is not exhausted by the sub menu being added (and if so, reject the request with the corresponding result code).
- 4) Store the data provided within this RPC correlating with the appID.
- 5) Add the sub menu with requested parameters (name, position) to the top level in-application menu.
- 6) Provide the response correspondingly to the result of RPC execution.

Note:

The applicable to this RPC result codes are provided in section 7.8.3 Response.

- 7) Provide the possibility for the User to enter sub menu added (empty by default) and choose among its commands (once added by SDL via UI.AddCommand).

Important Note:

Once the sub menus are added for the named application, they must remain accessible for the User (until UI.DeleteSubMenu comes) anytime the user opens application layout.

7.8.2.2 Parameters

Param Name	Type	Mandatory	Additional	Description
menuID	Integer	true	Minvalue = 1 Maxvalue = 2000000000	ID of the sub menu to be added. It is unique within the application concerned.
menuParams	Common.MenuParams	true	-	The name of the sub-menu to be added and its positions within menu items. See MenuParams.

appID	Integer	true	-	ID of the application requested this RPC.
-------	---------	------	---	---

7.8.2.3 MenuParams

Param Name	Type	Mandatory	Additional	Description
position	Integer	false	minvalue = 0 maxvalue = 1000	<p>This value is the position within the elements of top-level application menu where the sub-menu must be added to:</p> <ul style="list-style-type: none"> - If 0, the item must be inserted to the first position. - If 1, the item must be inserted to the second position. - Etc. <p>If the next sub menu comes with the same position, it must be added as the next item within corresponding 'position block'. (e.g. the sub menu with position 0 must be added</p> <ul style="list-style-type: none"> - after the previous item with position 0 - and before the first item with position 1). <p>If the value is greater than or equal to the number of elements of the top-level application menu, the sub menu must be appended to the end of the list.</p> <p>If omitted the entry must be added to the end of the list.</p>
menuName	String	true	maxlength = 500	The text that must be shown as a name of a sub - menu.

7.8.3 Response

Note:

There is a difference in message type for WebSocket and D-Bus connection (see the table below).

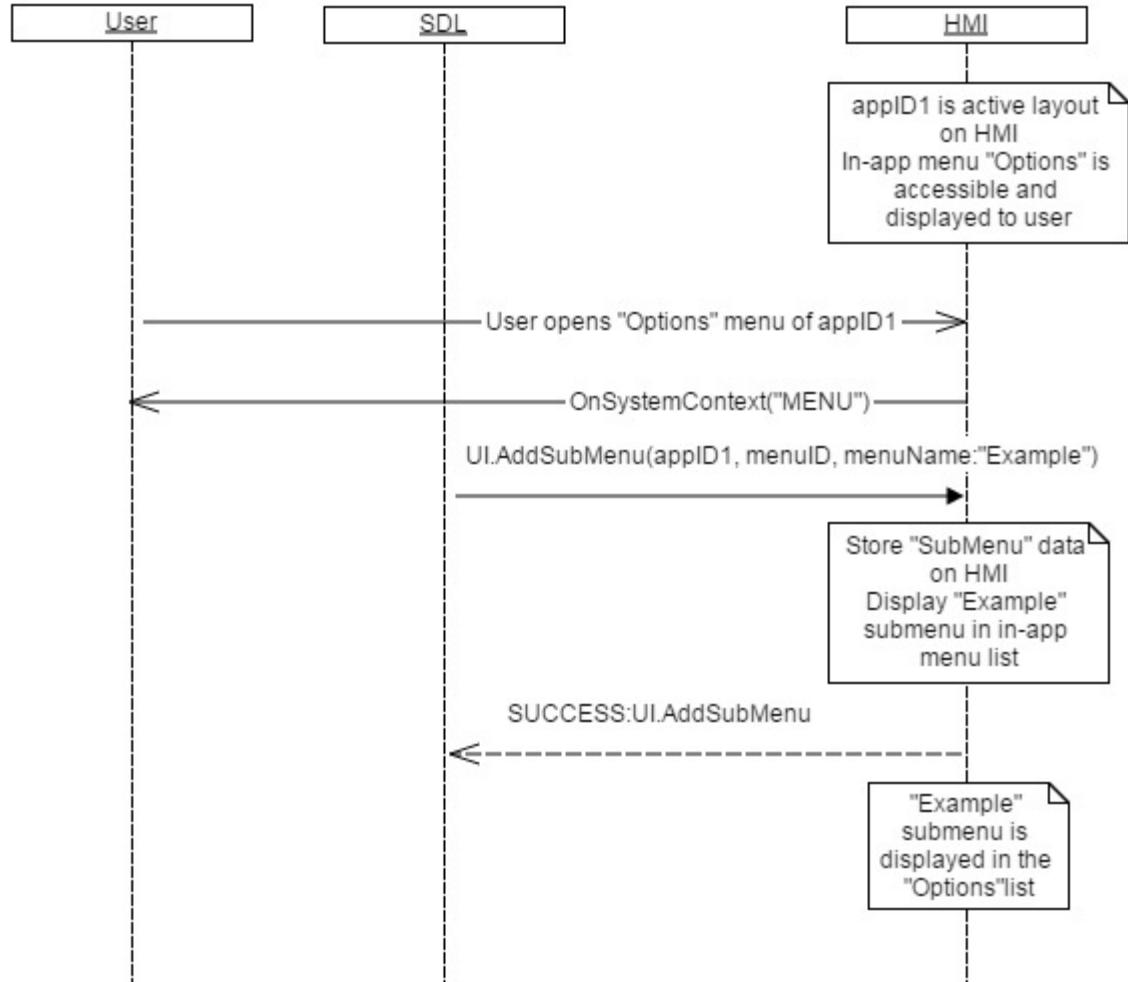
Result	Description	Message type		Message Params	Notes
		WebSocket	D-Bus		
Success	HMI has stored the data correlating it with appID and added the sub menu with the corresponding parameters to the in-application main menu.	JSON response	Method return	code: 0	
	DUPLICATE_NAME The submenu with the same menuName is already registered.			Code: 14	
Failure	REJECTED The limit of position items of the top level menu is exhausted.	JSON error message	Method return	Code: 4	Applicable for this RPC result codes. Please see Result Enumeration for
	INVALID_DATA The data sent is invalid (invalid JSON syntax, parameters out of bounds or of wrong type)			code: 11	

	<p>INVALID_ID The sub menu with requested menuID is already added for the named application. The app with appId in the request isn't registered</p> <p>GENERIC_ERROR: 1) The unknown issue occurred or other codes are not applicable.</p>		<p>code: 13</p>	<p>all SDL-recognized codes.</p>
			<p>code: 22</p>	

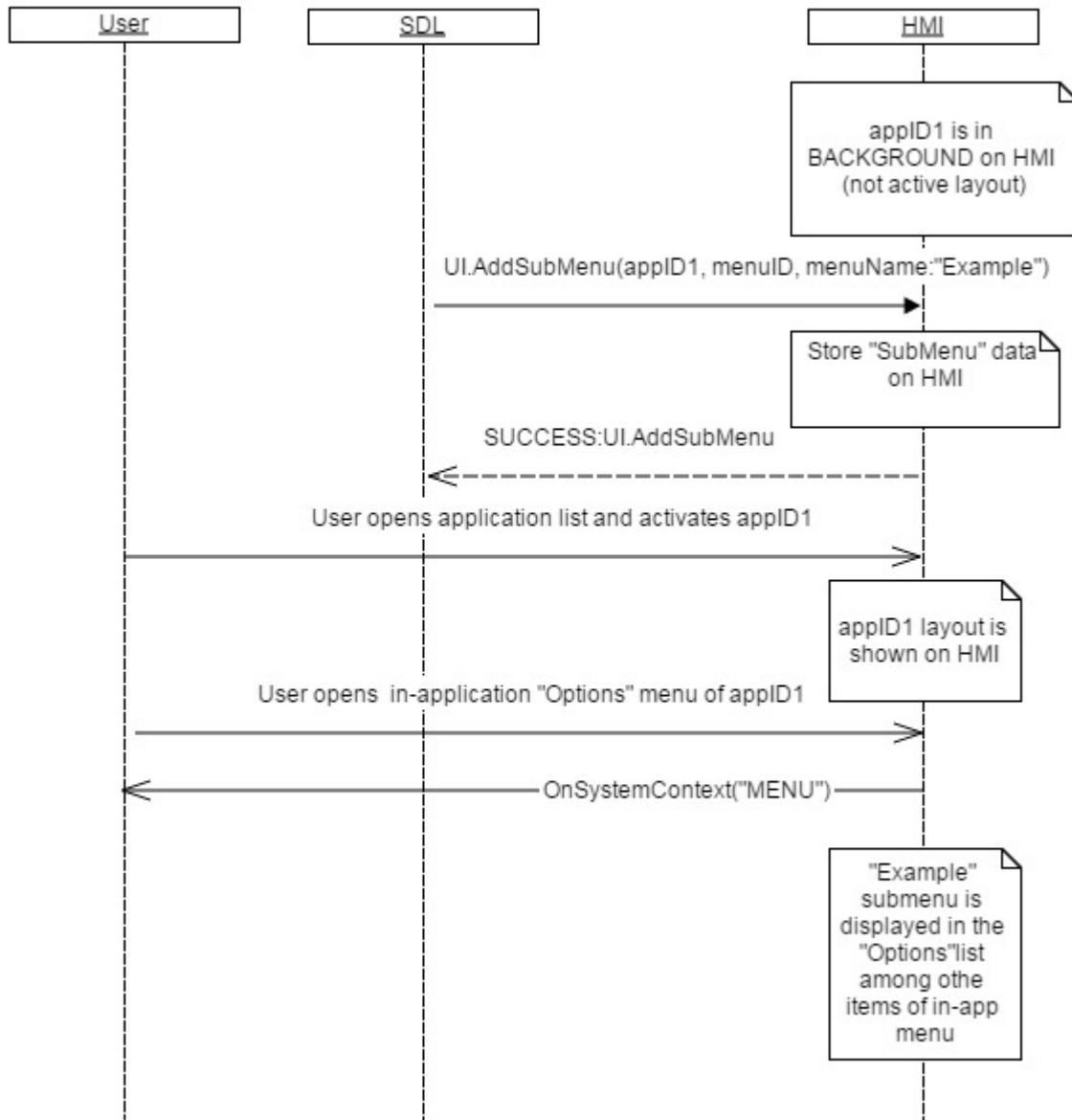
SDL Note: In case HMI does not respond SDL's request during SDL-default timeout (10 sec), SDL will return GENERIC_ERROR result code to the corresponding mobile app's request.

7.8.4 Sequence Diagrams

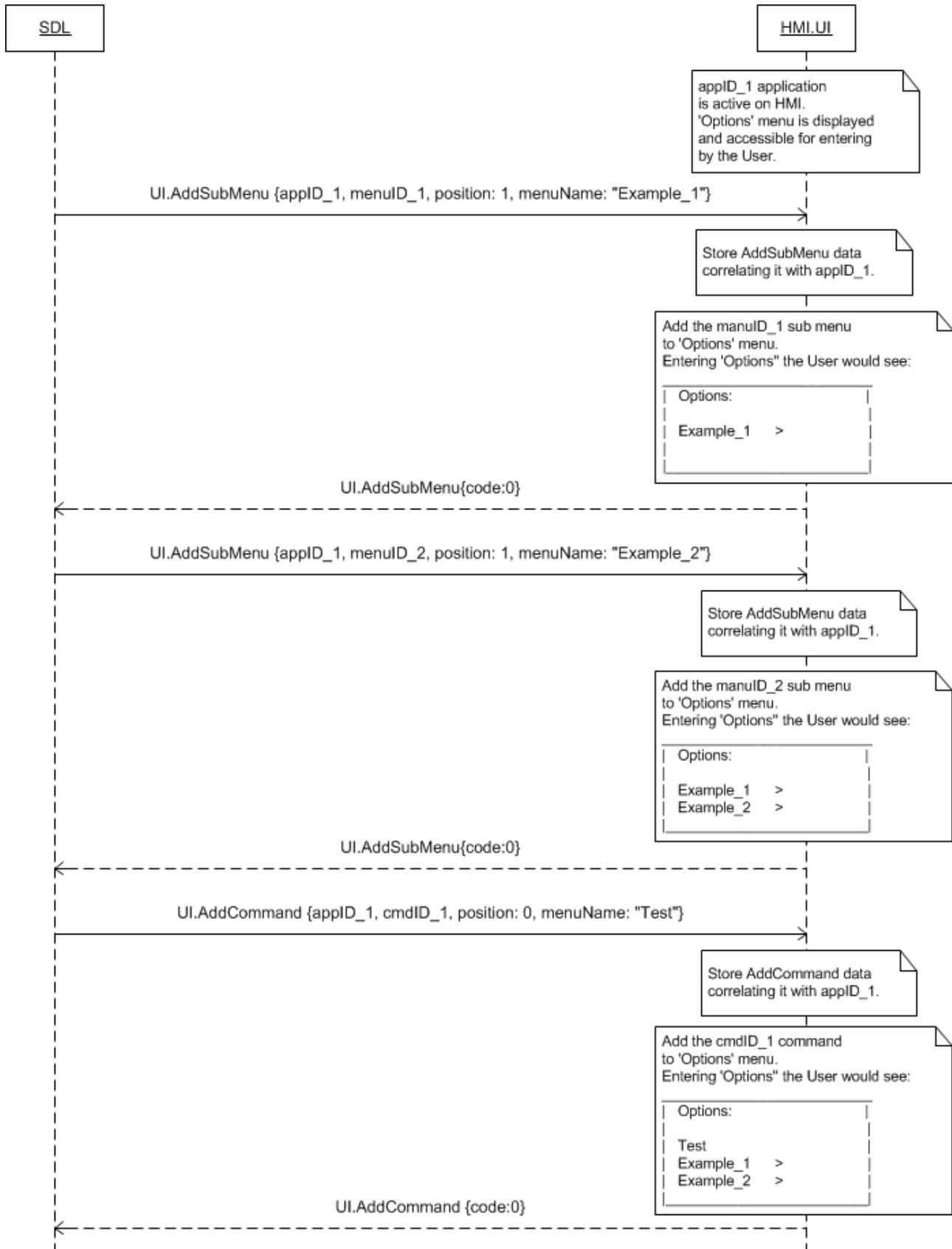
7.8.4.1 AddSubMenu for the application active on HMI



7.8.4.2 AddSubMenu for the application not active on HMI



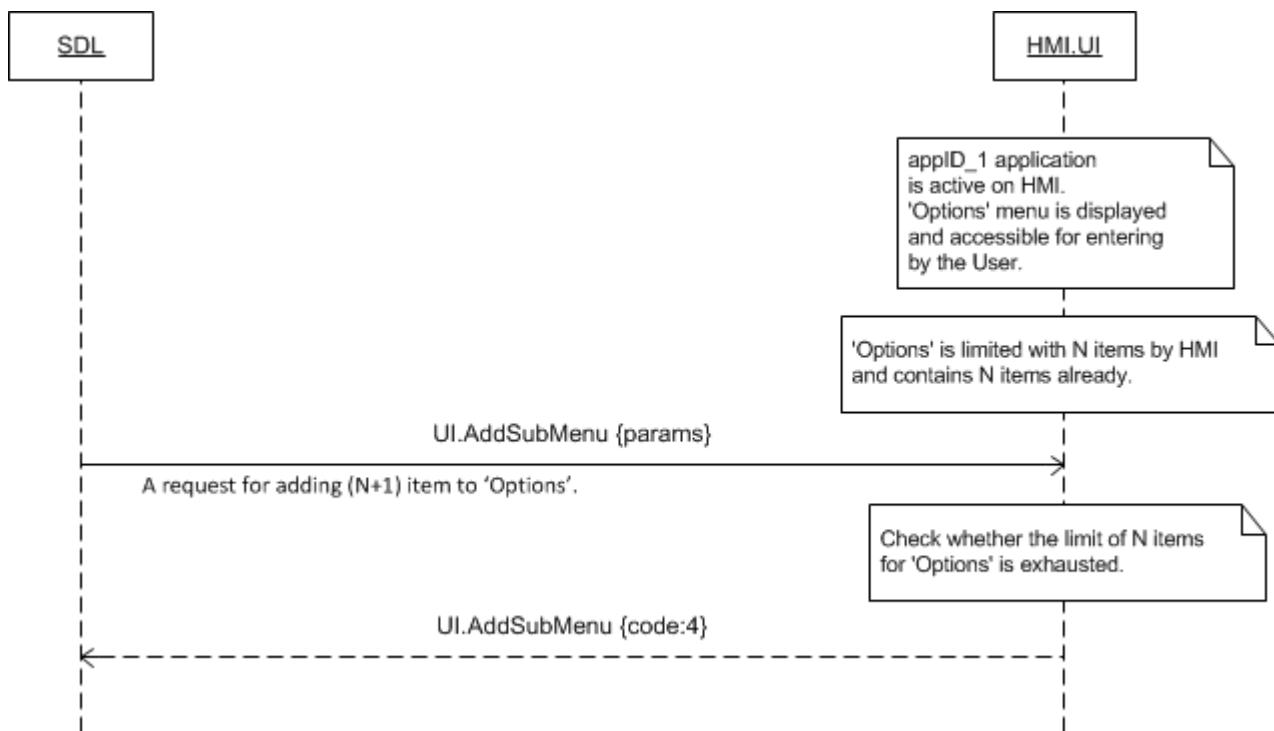
7.8.4.3 AddSubMenu: expected behavior of adding sub menus depending on position parameter



Note:

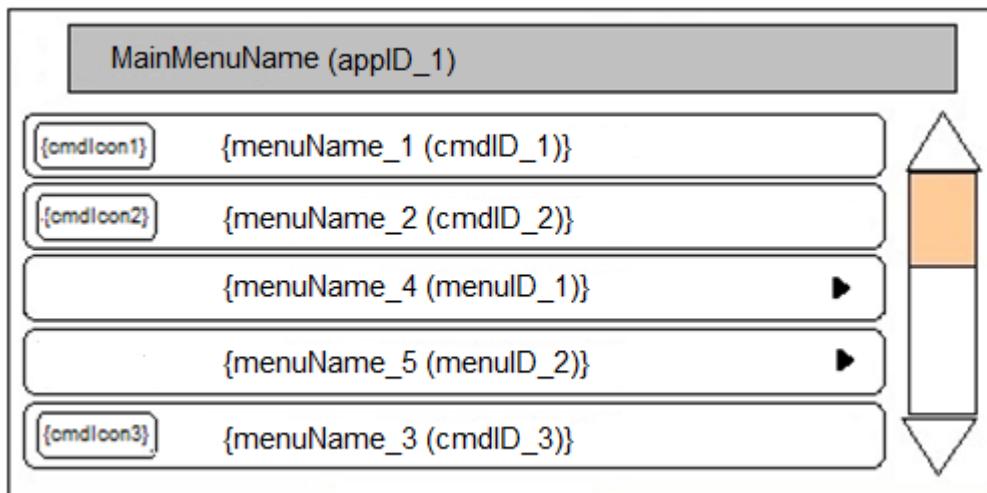
HMI should sort the in-application main menu items by position parameter independently of their type (sub menu/command).

7.8.4.4 AddSubMenu rejected because of the limit of menu items exhausted



7.8.5 Possible Layout

7.8.5.1 Application main menu with sub menus and commands



7.8.6 JSON Messages Examples

7.8.6.1 Request

```
{
  "id" : 112,
  "jsonrpc" : "2.0",
```

```

    "method" : "UI.AddSubMenu",
    "params" :
    {
        "menuID" : 345,
        "menuParams" :
        {
            "position" : 2,
            "menuName" : "Settings"
        },
        "appID" : 65464
    }
}

```

7.8.6.2 Response

```

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

```

7.8.6.3 Error message

```

{
    "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"
        }
    }
}

```

7.9 DeleteSubMenu

7.9.1 Description

Type:	Function
Sender:	SDL
Purpose:	Delete the sub menu from in-application menu.

SDL requests to delete the submenu with all related commands from in-application which was previously added via UI.AddSubMenu.

The request may arrive in both cases of activated and deactivated application on HMI (depends on Policy Table permissions, by default allowed to operate in all HMI levels except of NONE).

7.9.2 Request

7.9.2.1 Behavior

HMI must:

1. Check whether the named sub menu is currently open on UI and:

- If so, NOT delete this sub menu responding with appropriate result code (see IN_USE code, [section 7.9.3](#)).
- If not, continue processing the request.

2. Update the application's submenu data correspondingly.

3. Delete

- The sub menu identified with `menuID`.
- The commands that are related to this sub menu (which have `parentID` equal to `menuID` value)

4. Display updates after the main menu is opened on UI upon User's request:

- If the RPC arrived when the named application was active and had another menu or persistent display visible on UI
- If the RPC arrived when the named application was not active on HMI.

5. Provide the response corresponding to the result of RPC execution.

Note:

The applicable to this RPC result codes are provided in section 7.9.3 Response.

SDL note: In case the submenu contains the commands, SDL will first receive a SUCCESS response on `DeleteSubMenu` request and only then send `DeleteCommand` for all commands(VR and UI) related to the appropriate submenu. For more details see also [7.10.4.1 DeleteSubMenu which contains the commands](#)

Note:

- The value of `menuID` is previously sent via `AddSubMenu`.
- The application to which the request relates to, is identified by `appId` previously sent via `UpdateAppList` or `OnAppRegistered`.

7.9.2.2 Parameters

Param Name	Type	Mandatory	Additional	Description
<code>menuID</code>	Integer	true	Minvalue = 1 Maxvalue = 2000000000	ID that identifies the submenu to be deleted (the one that was previously provided via <code>AddSubMenu</code>).

Param Name	Type	Mandatory	Additional	Description
appID	Integer	true	-	ID of the application that requested this RPC.

7.9.3 Response

Note:

There is a difference in message type for WebSocket and D-Bus connection (see the table below).

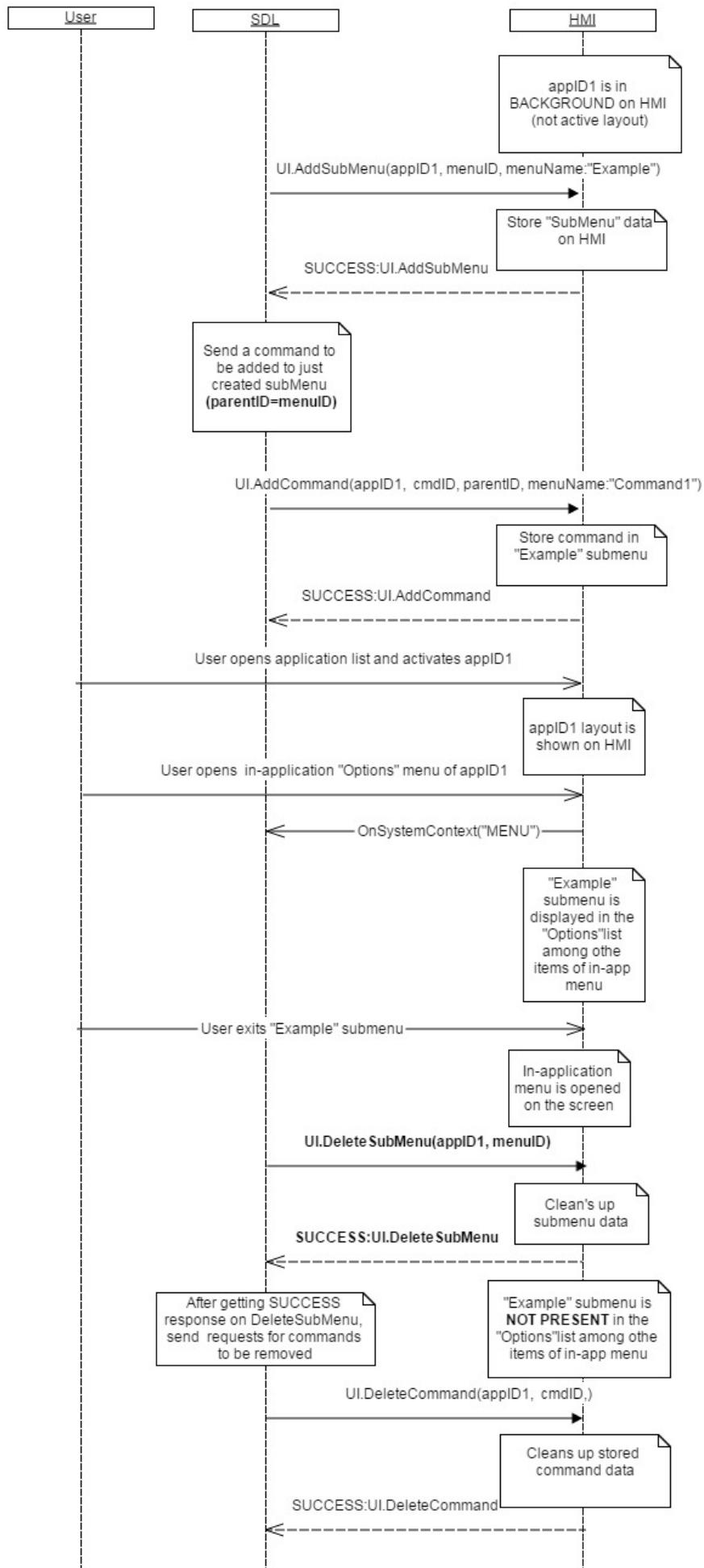
Result	Description	Message type		Message Params	Notes
		WebSocket	D-Bus		
Success	SUCCESS HMI has updated the named application related data and deleted the requested sub menu.	JSON response	Method return	code : 0	
Failure	IN_USE The named sub menu is currently open on UI. SubMenu must not be removed from in-application list	JSON error message	Method return	Code : 8	Applicable for this RPC result codes. Please see Result Enumeration for all SDL-recognized codes.
	INVALID_DATA The data sent is invalid (invalid JSON syntax, parameters are out of bounds or of wrong type)			code : 11	
	INVALID_ID The sub menu with requested menuID does not exist on HMI for the named application. applID provided is			Code : 13	

	not valid.			
	GENERIC_ERROR: 1) The unknown issue occurred or other codes are not applicable.		code : 22	

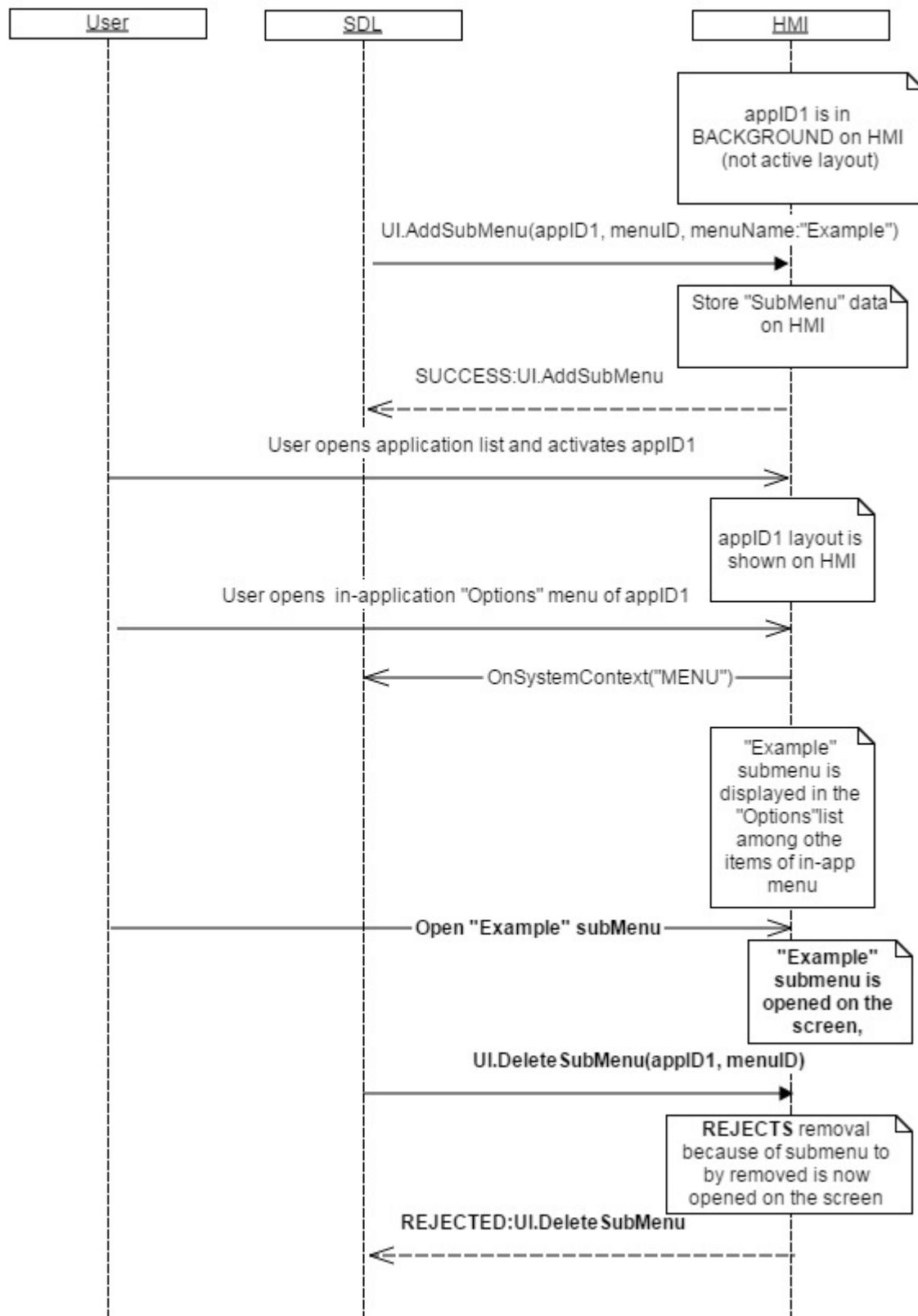
Note: In case HMI does not respond SDL's request during SDL-default timeout (10 sec), SDL will return GENERIC_ERROR result code to the corresponding mobile app's request.

7.9.4 Sequence Diagrams

7.9.4.1 DeleteSubmenu which contains the commands



**7.9.4.2 DeleteSubMenu which is now opened
on the screen**



7.9.5 JSON Messages Examples

7.9.5.1 Request

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

7.9.5.2 Response

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

7.9.5.3 Error message

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

7.10 UI.PerformInteraction

7.10.1 Description

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

SDL uses PerformInteraction for requesting User to make some choice or keyboard input, for example:

- To type in some data via keyboard which will be transferred on mobile side
- To make a choice between several suggested ones.

The request concerns the application being currently in focus on HMI.

Notes about PerformInteraction

1. *SDL may intend to perform one of three modes of interaction:*
 - 1.1. ‘Manual only’ – *when the User is expected to make a choice manually only (display, buttons, soft buttons are involved).*
 - 1.2. ‘VR only’ – *when the User is expected to make a choice by voice only (voice recognition module is involved).*
 - 1.3. ‘Both’ – *when the User is expected to make a choice whichever by voice or manually (display, buttons, soft buttons, VR module are involved).*
2. *SDL sends both VR.PerformInteraction and UI.PerformInteraction for each of above types of interaction. HMI’s expected behavior differs depending on interaction mode (see section 7.12.2.1 Behavior).*
3. *SDL provides HMI with the set(s) of choices to be presented to the User. See the below Notes of CreateInteractionChoiceSet.*

Notes about CreateInteractionChoiceSet:

1. *Mobile application creates a set (or several sets) of UI choices which are stored on until SDL sends UI choices (those to be displayed) within UI.PerformInteraction directly.*
2. *VRSynonyms set (or several sets) are provided to HMI by SDL in advance, VR synonyms of created choices must be sent via VR.AddCommand (see [section 9.4](#) for details). They are then referenced in VR.PerformInteraction (see [section 9.12](#)).*

7.10.2 Request

7.10.2.1 Behavior

HMI must:

1. Distinguish the interaction mode by the following characteristics:

Mode	SDL sends both RPCs with the following parameters:	Note
VR_ONLY	VR.PerformInteraction {grammarID, params}	grammarID is present
	UI.PerformInteraction {params}	ChoiceSet or layoutMode MUST NOT be sent
MANUAL_ONLY	VR.PerformInteraction {params}	May be not sent at all to HMI, otherwise no grammarID must be sent
	UI.PerformInteraction {choiceSet, params}	choiceSet is present, otherwise InteractionLayout value MUST BE KEYBOARD
BOTH	VR.PerformInteraction {grammarID, params}	grammarID is present
	UI.PerformInteraction {choiceSet, params}	choiceSet is present

I. 'VR only' interaction mode:

HMI must:

1. Display the help dialog with the information provided by:
 - vrHelpTitle parameter – as a title of the message
 - vrHelp parameter – as a list of commands accessible for voice recognition.

Note:

In 'VR only' interaction mode HMI must display the help information only.

2. Respond with `SUCCESS` result code right after user has spoken a command and VR module recognized it (for other applicable result codes please see section 7.10.3 Response).

3. Close the UI help dialog upon the corresponding `VR.PerformInteraction` completes (by either the User's choice, or by VR timeout, or being aborted by system event of a higher priority).

Note:

Please see diagram in section 7.10.4.1

UI.PerformInteraction in 'VR only' mode.

Important: *SDL's timeout isn't applicable for VR_ONLY mode.*

II. 'Manual only' interaction mode:

1. Start timeout provided within `timeout` parameter.
2. Display the `PerformInteraction` dialog which layout must depend on `interactionLayout` parameter:

2.1. `ICON_ONLY` layout:

2.1.a) Use value of `initialText` parameter – to display the text on the first line of the `Perform Interaction` layout(mostly used as a prompting to the User to start an interaction).

2.1.b) Use values of `choiceSet` parameter – to display the choices-related information (text and/or image) in the form of icons.

2.1.c) Proceed with step 3. when the User clicks one of the choice-icons.

2.2. `ICON_WITH_SEARCH` layout:

2.2.a) Use value of `initialText` parameter – to display the text on the first line of the display (mostly used as a prompting to the User to start the interaction).

2.2.b) Use values of `choiceSet` parameter – to display the choices-related information (text and/or image) in the form of icons.

2.2.c) Display a 'Search tab' together with 'Search' button in the top of the dialog.

- 2.2.c.1) Display the touchscreen keyboard overlaying both choices and text when the user clicks the 'Search tab' area.
- 2.2.c.2) Provide OnKeyboardInput notification(s) (the sequence depends on KeyboardProperties defined by SDL within UI.SetGlobalProperties. Please see [section 7.25 OnKeyboardInput](#) for description and diagrams).
- 2.2.c.3) Proceed with step 3. When the User presses the 'Search' button.
- 2.2.d) Proceed with step 3. when the User clicks one of the choices in the list.

2.3. LIST_ONLY layout:

- 2.3.a) Use value of initialText parameter – to display the text on the first line of the display (mostly used as a prompting to the User to start the interaction).
- 2.3.b) Use values of choiceSet parameter – to display the choices-related information (text and/or image) in the form of the list.
- 2.3.c) Proceed with step 3. when the User clicks one of the choices in the list.

2.4. LIST_WITH_SEARCH layout:

- 2.4.a) Use value of initialText parameter – to display the text on the first line of the display (mostly used as a prompting to the User to start the interaction).
- 2.4.b) Use values of choiceSet parameter – to display the choices-related information (text and/or image) in the form of the list.
- 2.4.c) Display a 'Search tab' together with 'Search' button in the top of the dialog.
 - 2.4.c.1) Display the touchscreen keyboard overlaying both choices and text when the user clicks the 'Search tab' area.
 - 2.4.c.2) Provide OnKeyboardInput notification(s) (the sequence depends on KeyboardProperties defined by SDL within UI.SetGlobalProperties. Please see [section 7.25 OnKeyboardInput](#) for description and diagrams).
 - 2.4.c.3) Proceed with step 3. when the User presses the 'Search' button.
- 2.4.d) Proceed with step 3. When the User clicks one of the choices in the list.

2.5. KEYBOARD layout:

- 2.5.a) Display the touchscreen keyboard with 'Search' button.
- 2.5.b) Provide OnKeyboardInput notification(s) (the sequence depends on KeyboardProperties defined by SDL within UI.SetGlobalProperties. Please see [section 7.25 OnKeyboardInput](#) for description and diagrams).
- 2.5.c) Proceed with step 3. when the User presses the 'Search' button.

3. Respond with `SUCCESS` result code after the User has made a choice AND provide the value of:

- 3.1. `choiceID` parameter for the steps of '2.1.c)', '2.2.d)', '2.3.c)', '2.4.d)'.
- 3.2. `manualTextEntry` parameter for the steps of '2.2.c.3)', '2.4.c.3)', '2.5.c'.

(for all of applicable result codes please see section 7.10.3 Response).

III. 'Both' interaction mode:

1. Display the dialog with the information provided within:

- `vrHelpTitle` parameter – as a title of the message
- `vrHelp` parameter – as a list of commands accessible for voice recognition.

Note:

This 'VR Help'-dialog must be kept in correspondence with currently active VR session that SDL requests via `VR.PerformInteraction`.

2. Close 'VR Help'-dialog when the corresponding VR session:

- 2.1. Completes successfully by the User's VR choice. Proceed with **step 3**.
- 2.2. Completes by the timeout. Proceed with **step 4**.
- 2.3. Is aborted by the User. Proceed with **step 4**.

3. Respond with `SUCCESS` result code (for all of applicable result codes please see section 7.10.3 Response).

4. Start 'Manual only' interaction mode.

4.1. Start timeout provided within `timeout` parameter.

4.2. Display the `PerformInteraction` dialog which layout must depend on `interactionLayout` parameter:

4.2.1. `ICON_ONLY` layout:

4.2.1.a) Use value of `initialText` parameter – to display the text on the first line of the display (mostly used as a prompting to the User to start the interaction).

4.2.1.b) Use values of `choiceSet` parameter – to display the choices-related information (text and/or image) in the form of icons.

4.2.1.c) Proceed with step 4.3. when the User clicks one of the choice-icons.

4.2.2. `ICON_WITH_SEARCH` layout:

4.2.2.a) Use value of `initialText` parameter – to display the text on the first line of the display (mostly used as a

prompting to the User to start the interaction).

4.2.2.b) Use values of `choiceSet` parameter – to display the choices-related information (text and/or image) in the form of icons.

4.2.2.c) Display a ‘Search tab’ together with ‘Search’ button in the top of the dialog.

4.2.2.c.1) Display the touchscreen keyboard overlaying both choices and text when the user clicks the ‘Search tab’ area.

4.2.2.c.2) Provide `OnKeyboardInput` notification (s) (the sequence depends on `KeyboardProperties` defined by `SDL` within `UI.SetGlobalProperties`. Please see [section 7.25](#) [OnKeyboardInput](#) for description and diagrams).

4.2.2.c.3) Proceed with step 4.3. When the User presses the ‘Search’ button.

4.2.2.d) Proceed with step 4.3. when the User clicks one of the choices in the list.

4.2.3. LIST_ONLY layout:

4.2.3.a) Use value of `initialText` parameter – to display the text on the first line of the display (mostly used as a prompting to the User to start the interaction).

4.2.3.b) Use values of `choiceSet` parameter – to display the choices-related information (text and/or image) in the form of the list.

4.2.3.c) Proceed with step 4.3. when the User clicks one of the choices in the list.

4.2.4. LIST_WITH_SEARCH layout:

4.2.4.a) Use value of `initialText` parameter – to display the text on the first line of the display (mostly used as a prompting to the User to start the interaction).

4.2.4.b) Use values of `choiceSet` parameter – to display the choices-related information (text and/or image) in the form of the list.

4.2.4.c) Display a ‘Search tab’ together with ‘Search’ button in the top of the dialog.

4.2.4.c.1) Display the touchscreen keyboard overlaying both choices and text when the user clicks the ‘Search tab’ area.

4.2.4.c.2) Provide `OnKeyboardInput` notification(s)

(the sequence depends on KeyboardProperties defined by SDL within UI.SetGlobalProperties. Please see [section 7.25 OnKeyboardInput](#) for description and diagrams).

4.2.4.c.3) Proceed with step 4.3. when the User presses the 'Search' button.

4.2.4.d) Proceed with step 4.3. When the User clicks one of the choices in the list.

4.2.5. KEYBOARD layout:

4.2.5.a) Display the touchscreen keyboard with 'Search' button.

4.2.5.b) Provide OnKeyboardInput notification(s) (the sequence depends on KeyboardProperties defined by SDL within UI.SetGlobalProperties. Please see [section 7.25 OnKeyboardInput](#) for description and diagrams).

4.2.5.c) Proceed with step 4.3. when the User presses the 'Search' button.

4.3. Respond with SUCCESS result code after the User has made a choice AND provide the value of:

4.3.1. choiceID parameter for the steps of '4.2.1.c)', '4.2.2.d)', '4.2.3.c)', '4.2.4.d)'.

4.3.2. manualTextEntry parameter for the steps of '4.2.2.c.3)', '4.2.4.c.3)', '4.2.5.c)'.

(for all of applicable result codes please see section 7.12.3 Response).

7.10.2.2 Parameters

Param Name	Type	Mandatory	Additional	Description
initialText	Common.TextFieldStruct	false	-	<p>The text that must be displayed:</p> <ul style="list-style-type: none"> - When the interaction begins - On the first line of a multiline screen - Centered. <p>See TextFieldStruct, TextFieldName (uses initialInteractionText).</p>
choiceSet	Common.Choice	false	Array = true Minsize = 1 Maxsize = 100	<p>Array of one or more Choice Sets. Each one is represented with:</p> <ul style="list-style-type: none"> - The mandatory ID (to be returned by HMI in the response if the corresponding Choice Set is selected by the User) - Optional name and/or picture. <p>See Choice.</p> <p>HMI must provide the possibility for the User to choose among these choices.</p>

vrHelpTitle	String	false	Maxlength = 500	VR Help Title text. If omitted on supported displays, the default HU system help title should be used.
vrHelp	Common.VrHelpItem	false	Array = true Minsize = 1 Maxsize = 100	VR Help Items that must be displayed on-screen after HMI recognizes 'Help' command during PerformInteraction. See VrHelpItem.
timeout	Integer	true	Minvalue = 5000 Maxvalue = 100000 Defvalue = 10000	Timeout in milliseconds. The amount of time the HMI must wait for the User to make a choice. After the time is up HMI respond with TIMED_OUT result code.
interactionLayout	Common.LayoutMode	false	-	See LayoutMode.
appID	Integer	true	-	ID of the application related to this RPC.

7.10.2.3 TextFieldStruct Structure

Param Name	Type	Mandatory	Additional	Description
fieldName	Common.TextFieldIdName	true	-	The name of the field where the text must be displayed in.
fieldText	String	true	Maxlength = 500	The text to be displayed.

7.10.2.4 TextFieldName

Element name	Short Description
initialInteractionText	Must be displayed when the interaction begins. The text must be displayed on the first line of a multiline display, and must be centered. Applies to PerformInteraction

7.10.2.5 Choice

Param Name	Type	Mandatory	Additional	Description
choiceID	Integer	true	minvalue = 0 maxvalue = 65535	The unique within the concerned application.

Param Name	Type	Mandatory	Additional	Description
				an identifier for this choice
menuName	String	false	MaxLength = 500	The text to be displayed in the onscreen menu indicating the name of the choice (e.g. 'Yes').
image	Common.Image	false	-	Image that must appear in the menu, representing this choice. See Image
secondaryText	String	false	MaxLength = 500	Optional secondary text to display; e.g. address of POI in a search result entry
tertiaryText	String	false	MaxLength = 500	Optional tertiary text to display; e.g. distance to POI for a search result entry
secondaryImage	Common.Image	false	-	Optional secondary image struct for

Param Name	Type	Mandatory	Additional	Description
				choice

7.10.2.6 VrHelpItem

Param Name	Type	Mandatory	Additional	Description
text	String	True	maxlength = 500	The text that must be displayed as a name for VR Help item.
image	Common.Image	False	-	Image that must be displayed by HU to represent the VR Help item. See Image
position	Integer	false	minvalue = 1 maxvalue = 100	This value is the position within the elements of VR Help menu where the item must be added to:

7.10.2.7 LayoutMode

Element name	Short Description
ICON_ONLY	This mode causes the interaction to display a set of choices as icons during PerformInteraction running
ICON_WITH_SEARCH	This mode causes the interaction to display the previous set of choices as icons along with a search field on HMI.
LIST_ONLY	This mode causes the interaction to display the set of choices as a list of icons during PerformInteraction running
LIST_WITH_SEARCH	This mode causes the interaction to display the previous set of choices as a list along with a search field on HMI.
KEYBOARD	This mode causes the interaction to immediately display a keyboard entry on HMI.

7.10.3 Response

Note:

There is a difference in message type for WebSocket and D-Bus connection (see the table below).

Result	Description	Message type	Message	Notes
--------	-------------	--------------	---------	-------

		WebSocket	D-Bus	Params	
Success	SUCCESS The cases described in 7.10.2: 'I.2', 'II.3', 'III.3', 'III.4.3'.	JSON response	Regular response	choiceID, manualTextEntry, code: 0	See section 7.10.4 Parameters.
	INVALID_DATA The data sent is invalid (invalid JSON syntax, parameters are out of bounds or of wrong type)			code: 11	
	INVALID_ID ChoiceID or appId is invalid (e.g. doesn't exist)			code: 13	
	ABORTED The interaction is aborted by the User or system event of higher priority.			code: 5	
	TIMED_OUT The User has not made a choice during timeout defined within request.			code: 10	
	GENERIC_ERROR: 1) The unknown issue occurred or other codes are not applicable.			code: 22	

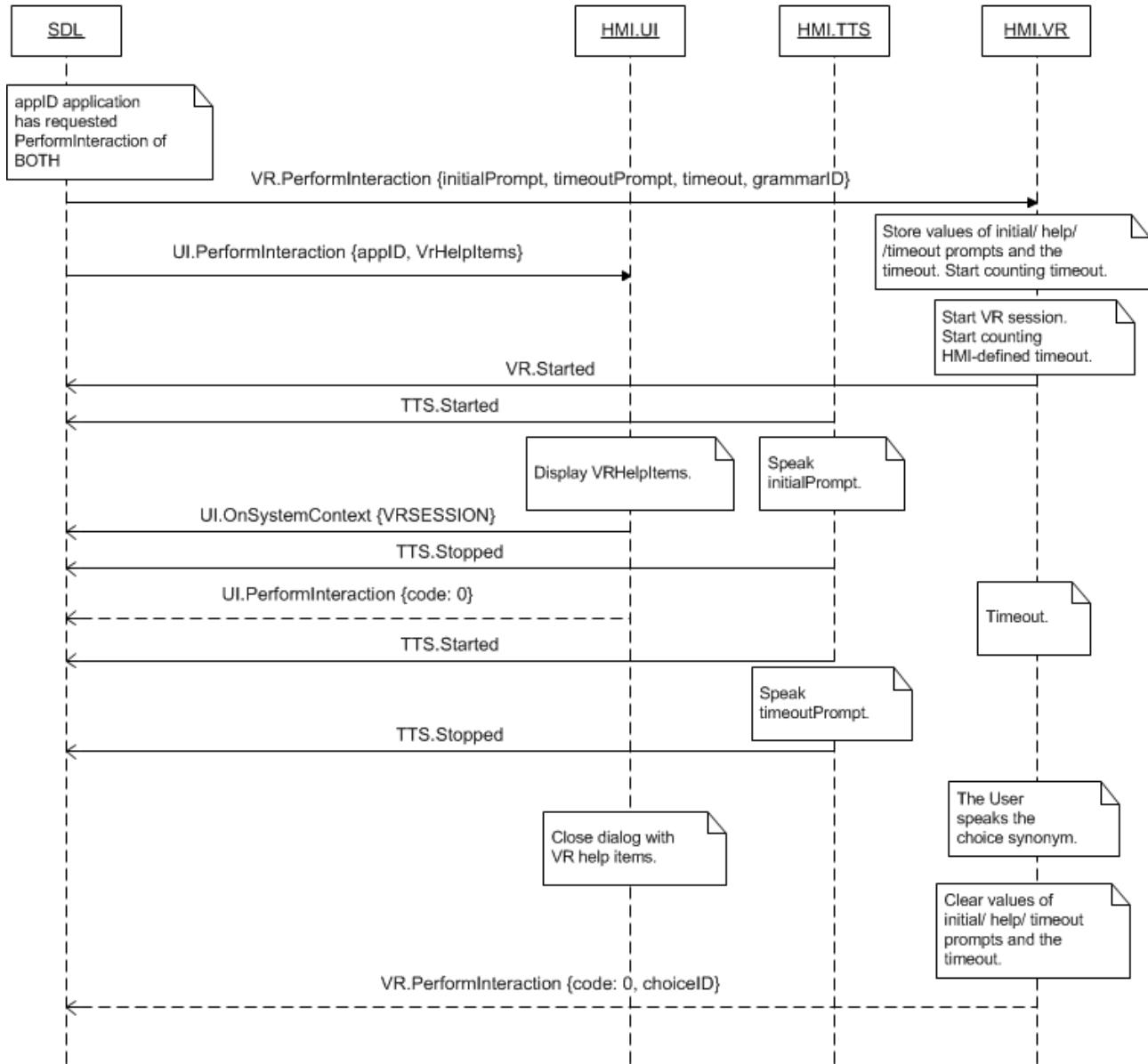
SDL Note: In case HMI does not respond SDL's request during timeout defined in a request, SDL will return GENERIC_ERROR result code to the corresponding mobile app's request.

7.11.3 Parameters

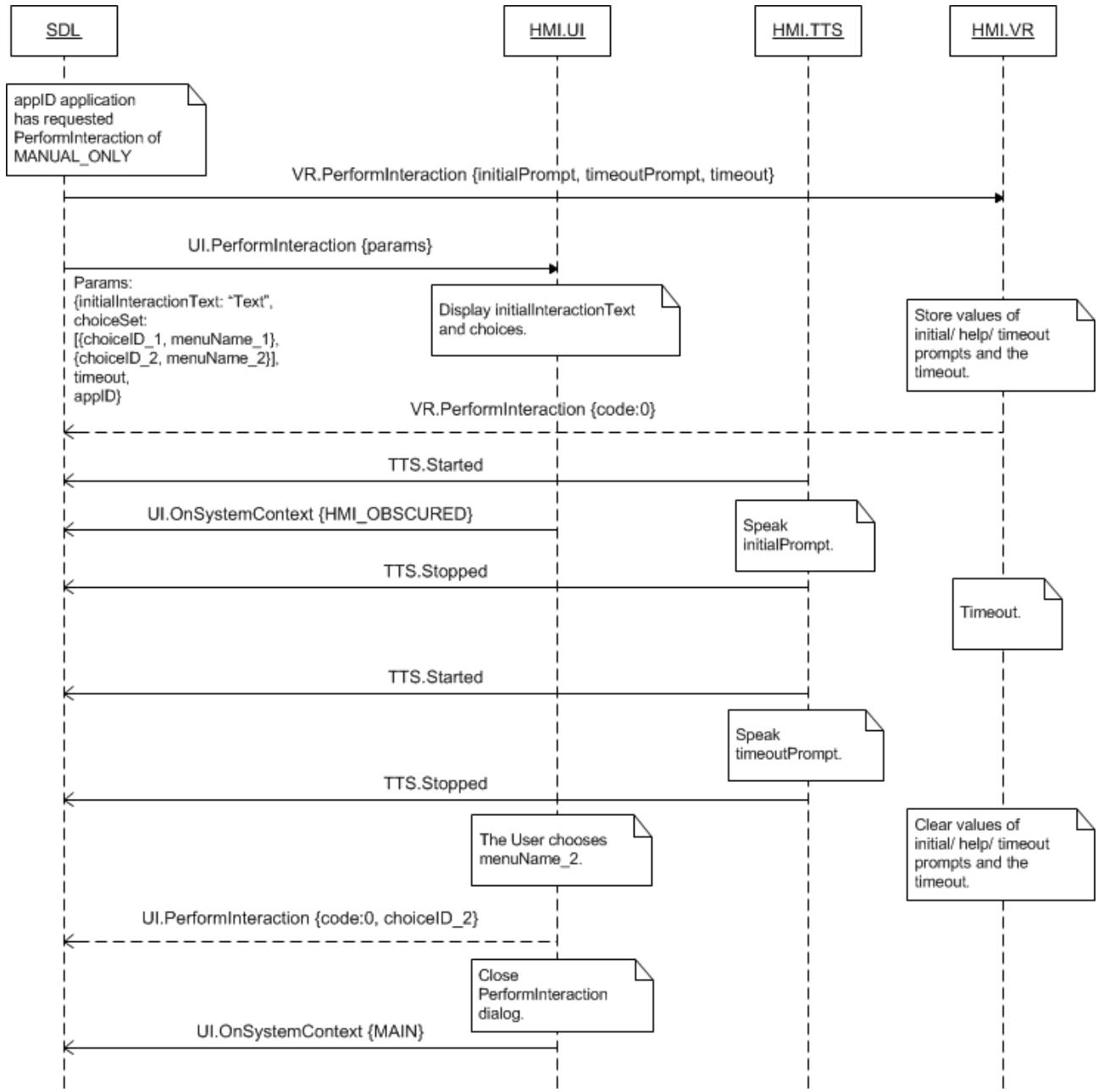
Param Name	Type	Mandatory	Additional	Description
choiceID	Integer	false	minvalue = 0 maxvalue = 2000000000	ID that represents the choice made by the User (one among those provided within the request).
manualTextEntry	String	false	minlength = 0 maxlength = 500	Manually entered text selection, e.g. through keyboard Can be returned in lieu of choiceID, depending on trigger source

7.10.4 Sequence Diagrams

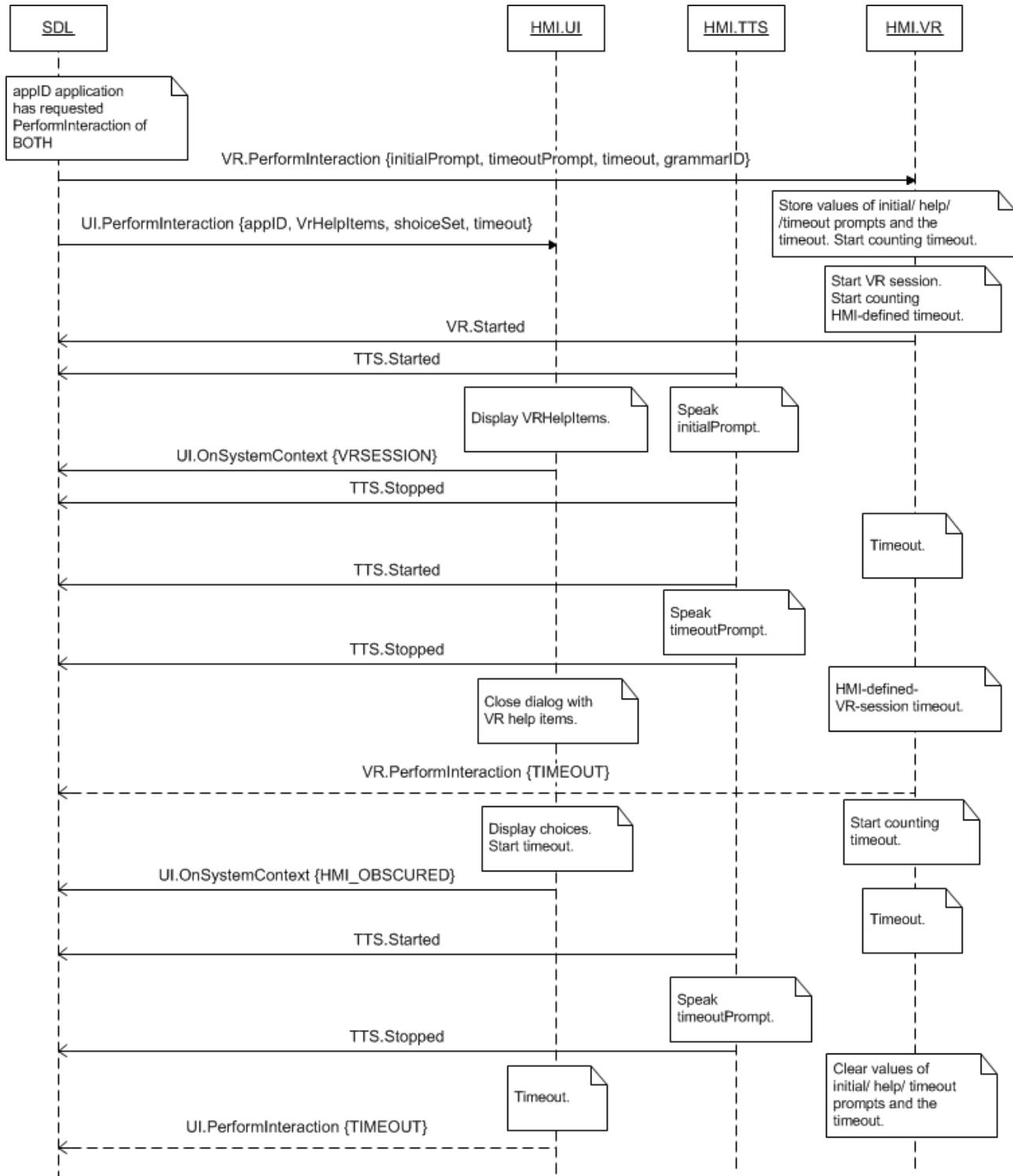
7.10.4.1 UI.PerformInteraction in 'VR only' mode successfully completed



7.10.4.2 UI.PerformInteraction in 'Manual only' mode successfully completed



7.10.4.3 UI.PerformInteraction in 'Both' mode timed out



7.10.5 JSON Messages Examples

7.10.5.1 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
                                }
                            ]
                        }
                    ]
                }
            ]
        }
    ]
}
```

```

        "position" : 3
    }
],
"timeout" : 15000,
"appID" : 6493
}
}
```

7.10.5.2 Response

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

7.10.5.3 Error message

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

7.11 SetMediaClockTimer

7.11.1 Description

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

With this RPC SDL requests to
- set-up the initial value for media clock timer
- update it correspondingly to the provided parameters of the request.

The request may concern the application not being currently active on HMI.

Note:

SDL uses the information about:
- *mediaClock field*

- the format of time value
confirmed to be supported by HMI within response to
`UI.GetCapabilities`
SDL will not send the request if `mediaClock` field is not
supported.

7.11.2 Request

7.11.2.1 Behavior

Important Note:

If the request comes for the application

- Currently active on HMI, the requested updates must be immediately displayed
- Currently NOT active (being in background) on HMI, HMI must
 - Perform the requested updates in background
 - Display the value of the media clock valid at the time the corresponding application becomes activated on HMI.

HMI must:

1. Perform the following depending on the `updateMode` parameter value:

- COUNTUP / COUNTDOWN modes:
 - Start counting up / down from the requested `startTime` value with the step of 1 second
 - Continue counting up / down until:
 - The next request of `SetMediaClockTimer` with appropriate parameters comes
 - Zero value in case of COUNTDOWN.
- PAUSE mode:
 - Pause the timer that is counting up / down
 - If `startTime` or `endTime` are provided, the values must be updated on HMI.
- RESUME mode:
 - Continue counting up / down (in the mode that was in effect before pausing) the clock that has been paused. In case `startTime` is provided in the request, timer must be updated with a new value
- CLEAR mode:
 - Clear the `startTime` value only in case `startTime` 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.

Important Note:

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.

Note:

Initially, the appID together with other application-related information is provided by SDL within one of UpdateAppList and OnAppRegistered RPCs.

2. Respond with the result code correspondingly to the results of this RPC execution.

Note:

The applicable to this RPC result codes are provided in section 7.11.3 Response.

Note:

The sequence diagrams describing the expected HMI behavior are provided in the section 7.11.4 Sequence Diagrams

7.11.2.2 Parameters

Param Name	Type	Mandatory	Description
startTime	Common.TimeFormat	false	The starting time value to be counted up / down from. See TimeFormat
updateMode	Common.ClockUpdateMode	true	The update method of the clock value: COUNTUP, COUNTDOWN, PAUSE, RESUME, CLEAR. In case of RESUME, or CLEAR, the startTime value must be ignored and left out if provided. See ClockUpdateMode.
appID	Integer	true	ID of the application that concerns this RPC.

7.11.2.3 TimeFormat

Param Name	Type	Mandatory	Additional	Description
hours	Integer	true	minvalue = 0 maxvalue = 59	The hour of the media clock.
minutes	Integer	true	minvalue = 0 maxvalue = 59	The minute value of media timer.
seconds	Integer	true	minvalue = 0 maxvalue = 59	The seconds value of media timer.

7.11.2.4 ClockUpdateMode

Element name	Short Description
COUNTUP	HMI must start the media clock timer counting upwards, as in time elapsed, in increments of 1 second.

Element name	Short Description
COUNTDOWN	HMI must start the media clock timer counting downwards, as in time remaining, in decrements of 1 second.
PAUSE	HMI must pause the media clock timer.
RESUME	HMI must resume the media clock timer. The timer must resume counting in whatever mode was in effect before pausing (i.e. COUNTUP or COUNTDOWN).
CLEAR	HMI must clear the media clock timer.

7.11.3 Response

Note:

There is a difference in message type for WebSocket and D-Bus connection (see the table below).

Result	Description	Message type		Message Params	Notes
		WebSocket	D-Bus		
Success	SUCCESS: 1) HMI has started counting up / down the media clock from the requested value (in case of COUNTUP / 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 down	JSON response	Method return	code : 0	HMI must respond right after the action requested with update Mode parameter has been performed. See section 7.11.4 Sequence Diagrams for details.

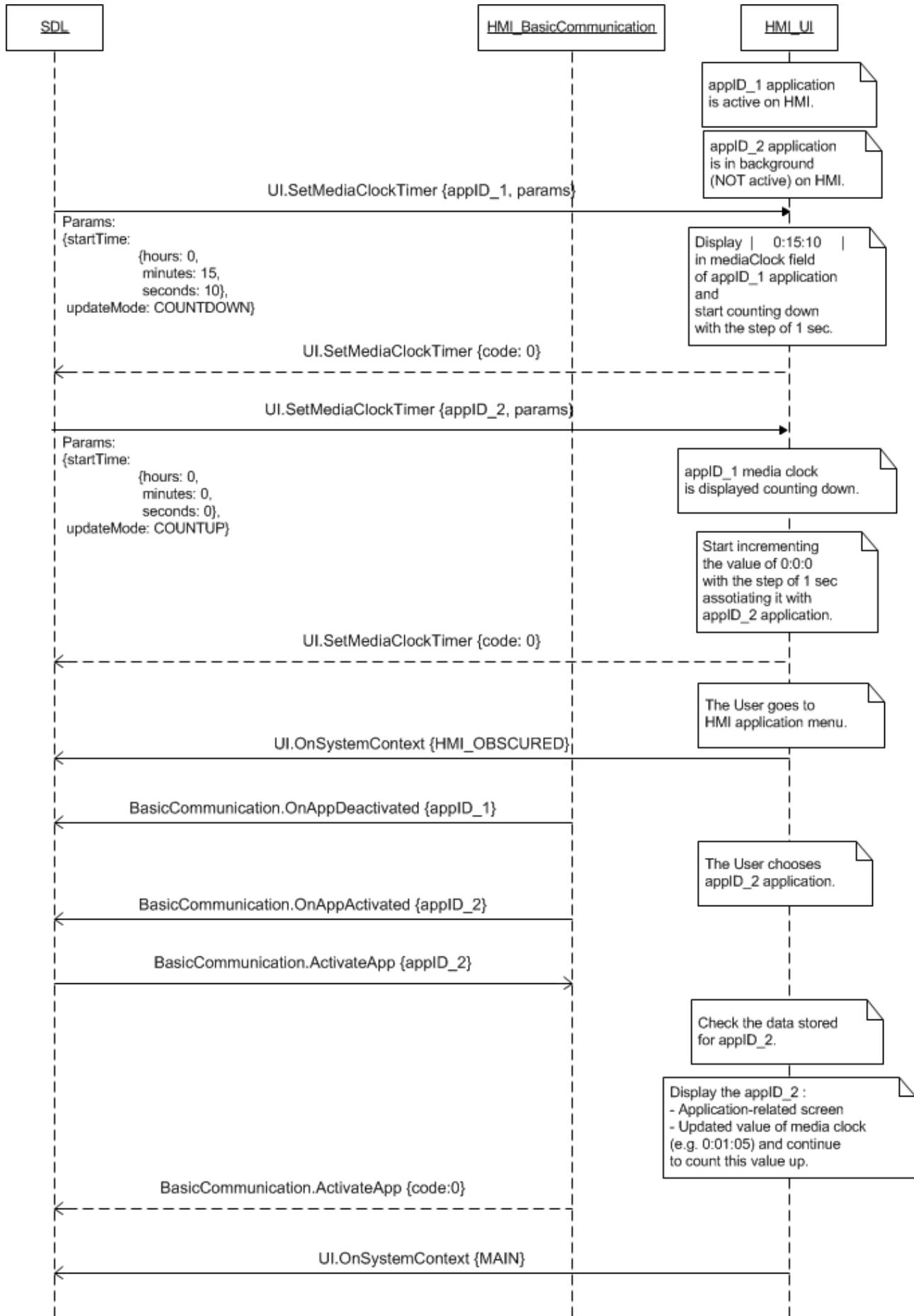
	(RESUME requested mode). 4) HMI has cleared the existing media clock value (CLEAR requested mode).				
Failure	IGNORED: 1) Request with RESUME mode arrives when the media clock is counting, already resumed or cleared with the previous request. 2) Request with PAUSE mode arrives when the media clock is paused or cleared with the previous request.	JSON error message	Method return	code : 6	Applicable to this RPC result codes.
	INVALID_DATA: The data sent is invalid (invalid JSON syntax or parameters out of bounds or of wrong type)			code : 11	Please see Result Enumeration for all SDL-supported codes.
	INVALID_ID appId is invalid (e.g. doesn't exist)			code : 13	
	GENERIC_ERROR: 1) The unknown issue occurred or			code : 22	

	other codes are not applicable.				
--	---------------------------------------	--	--	--	--

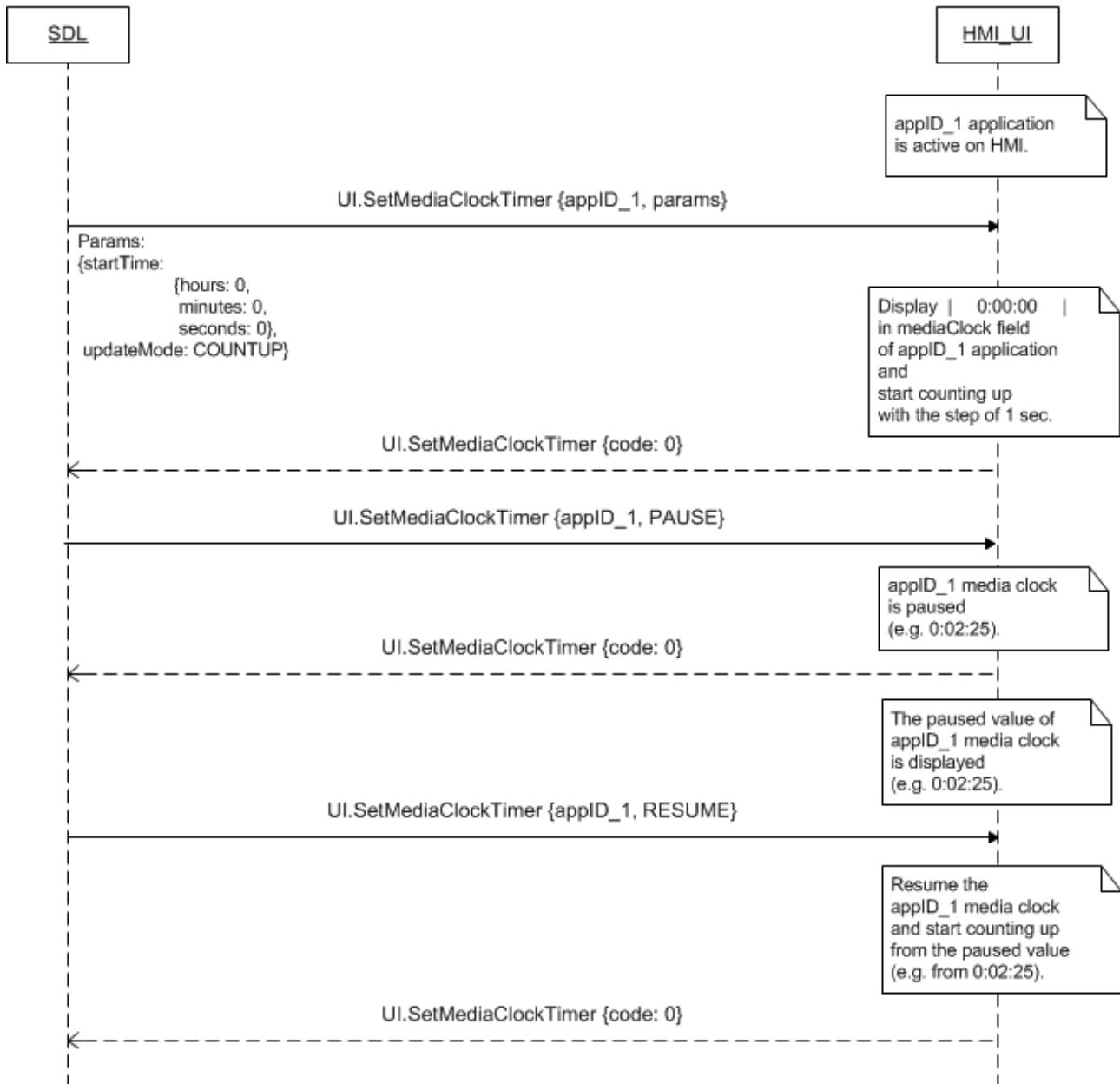
SDL Note: In case HMI does not respond SDL's request during SDL-default timeout (10 sec), SDL will return `GENERIC_ERROR` result code to the corresponding mobile app's request.

7.11.4 Sequence Diagrams

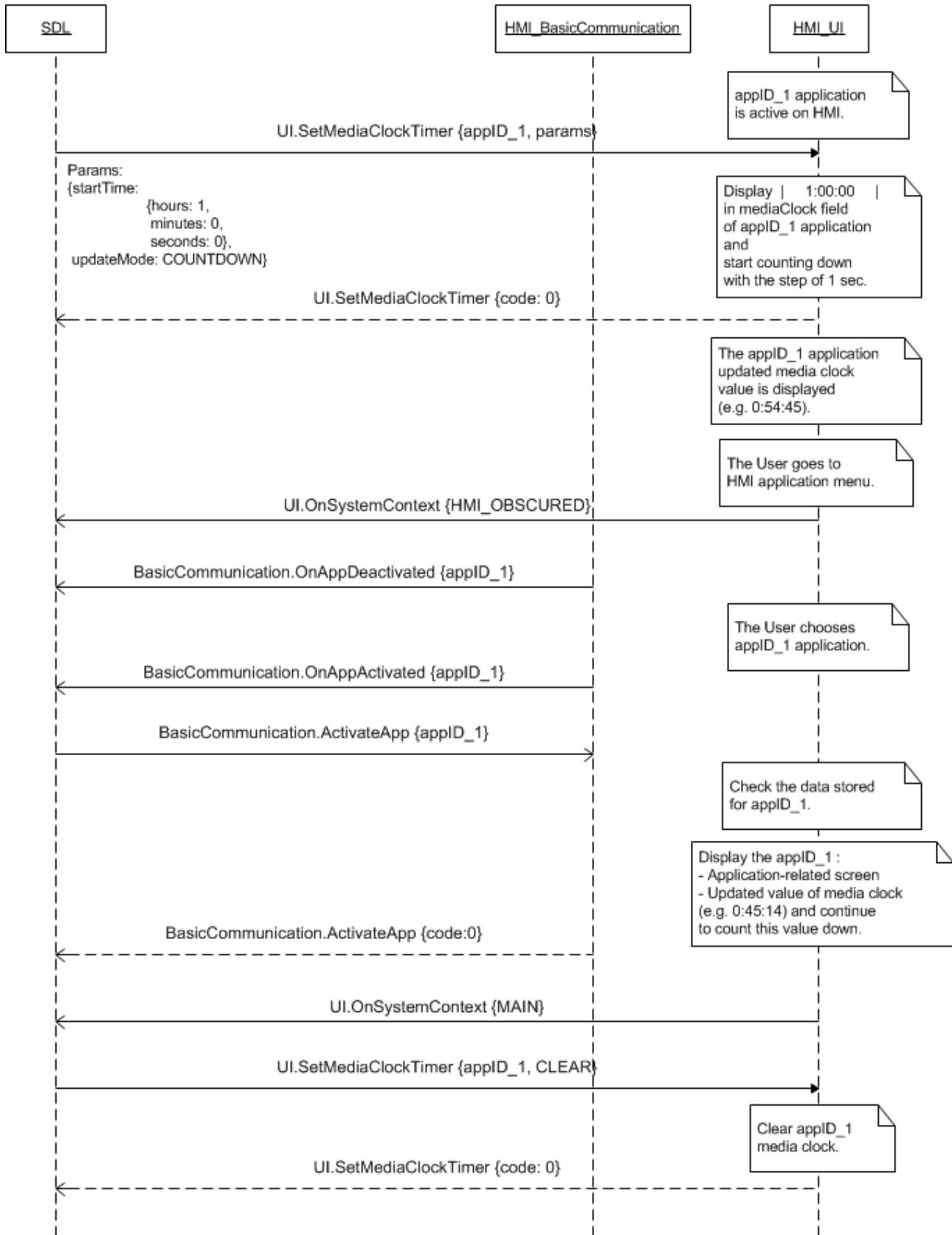
7.11.4.1 SetMediaClockTimer of COUNTUP and COUNTDOWN modes for the cases of active and background applications



7.11.4.2 SetMediaClockTimer of PAUSE and RESUME for the application active on HMI



**7.11.4.3 SetMediaClockTimer of COUNTDOWN
for the active application that is then
deactivated and activated again and
SetMediaClockTimer of CLEAR for the active
application.**



7.11.5 JSON Messages Examples

7.11.5.1 Request

```
{
  "id" : 109,
```

```

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

```

7.11.5.2 Response

```

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

```

7.11.5.3 Error message

```

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

```

7.12 SetGlobalProperties

7.12.1 Description

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

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 whatever 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).

7.12.2 Request

7.12.2.1 Behavior

HMI 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. Set the requested values for:

- VR help layout: whenever the User activates VR, HMI must 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).

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.*
- In-application menu: HMI must 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.
- Touchscreen keyboard: If supported, HMI must display the onscreen keyboard upon User's request. SDL provides the properties of this keyboard:

Note:

The values for in-application menu and touchscreen keyboard are allowed by SDL for navigation type of application only.

3. Respond to the request.

Note:

The applicable to this RPC result codes are provided in section 7.11.3 Response.

Note:

The sequence diagrams describing the expected HMI behavior are provided in the section 7.11.4 Sequence Diagrams

7.12.2.2 Parameters

Param Name	Type	Mandatory	Additional	Description
vrHelpTitle	String	false	Maxlength = 500	The text that must be displayed as a title of VR Help Menu.
vrHelp	Common.VrHelpItem	false	Array = true Minsize = 1 Maxsize = 100	VR help items – i.e. the text strings to be displayed, and when pronounced by the user the recognition of any of which must trigger the corresponding VR command. See VrHelpItem.
menuTitle	String	false	Maxlength = 500	Optional text to label an app menu button (for certain touchscreen platforms).
menuIcon	Common.Image	false	–	Optional icon to draw on an app menu button (for certain touchscreen platforms).
keyboardProperties	Common.KeyboardProperties	false	–	On-screen keyboard configuration (if available). See also 15.3.2.1 UI section of hmi_capabilities.json
appID	Integer	true	–	ID of the application that concerns this RPC.

7.12.2.3 VrHelpItem

Param Name	Type	Mandatory	Additional	Description
text	String	True	maxlength = 500	The text that must be displayed as a name for VR Help item.
image	Common.Image	False	–	Image that must be displayed by HU to represent the VR Help item. See Image
position	Integer	false	minvalue = 1 maxvalue = 100	This value is the position within the elements of VR Help menu where the item must be added to:

7.12.2.4 KeyboardProperties

Param Name	Type	Mandatory	Additional	Description
language	Common.Language	false	–	The keyboard language.
keyboardLayout	Common.KeyboardLayout	false	–	Desired keyboard layout.
sendDynamicEntry	Boolean	false	–	In this mode, all keypresses will be sent as they occur. If disabled, entire string of text will be returned only once submitted by user. If omitted, this value will be set to

				FALSE.
keypressMode	Common.KeypressMode	false	-	Desired keypress mode. If omitted, this value will be set to RESEND_CURRENT_ENTRY.
limitedCharacterList	String	false	Array = true maxlength = 1 minsize = 1 maxsize = 100	Array of keyboard characters to enable. All omitted characters will be greyed out (disabled) on the keyboard. If omitted, the entire keyboard will be enabled.
autoCompleteText	String	false	maxlength = 1000	Allows an app to prepopulate the text field with a suggested or completed entry as the user types

7.12.2.5 KeyboardLayout

Element name	Short Description
QWERTY	QWERTY layout (the name comes from the first six keys appearing on the top left letter row of the keyboard and read from left to right)
QWERTZ	QWERTZ layout (the name comes from the first six keys appearing on the top left letter row of the keyboard and read from left to right)
AZERTY	AZERTY layout (the name comes from the first six keys appearing on the top left letter row of the keyboard and read from left to right)

7.12.3 Response

Note:

There is a difference in message type for WebSocket and D-Bus connection (see the table below).

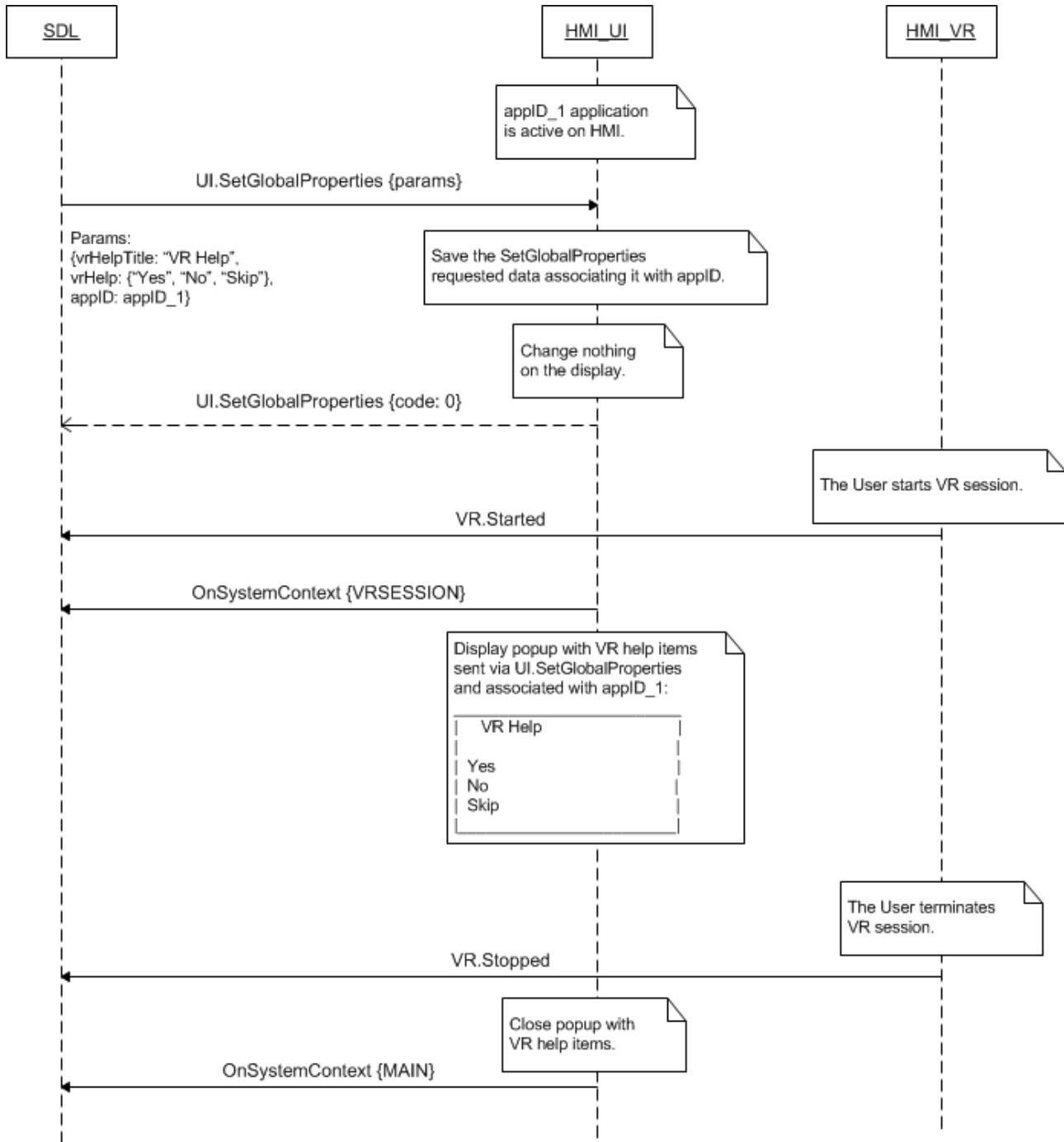
Result	Description	Message type		Message Params	Notes
		WebSocket	D-Bus		
Success	SUCCESS: HMI has set the requested properties..	JSON response	Method return	code : 0	
	INVALID_ID applID is not valid (e.g. doesn't exist)	JSON error message		code : 13	Applicable for this RPC result codes.
Failure	INVALID_DATA: The data sent is invalid (invalid		Method return	code : 11	Please see Result Enumer

	JSON syntax or parameters out of bounds or of wrong type)			ation for all SDL-supported codes.
	GENERIC_ERROR: 1) The unknown issue occurred or other codes are not applicable.		code : 22	

SDL 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.

7.12.4 Sequence Diagrams

7.12.4.1 SetGlobalProperties for the application active on HMI and the later VR activation



7.12.5 JSON Messages Examples

7.12.5.1 Request

```
{
  "id" : 116,
  "jsonrpc" : "2.0",
  "method" : "UI.SetGlobalProperties",
  "params" :
  {
    "vrHelpTitle" : "Choose the action",
    "vrHelp" :
    [
      "Yes"
    ]
  }
}
```

```
{
    "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
}
```

7.12.5.2 Response

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

7.12.5.3 Error message

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

7.13 ChangeRegistration

7.13.1 Description

Type:	Function
Sender:	SDL
Purpose:	Change the display language for the named application on HMI

SDL requests to set-up a display language for the named application on HMI.

The request may arrive for the application whatever 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).

Note:

*SDL will send the language value confirmed to be supported by HMI via UI.GetCapabilities.
HMI display language and application display language (requested via UI.ChangeRegistration) might be different.*

7.13.2 Request

7.13.2.1 Behavior

- 1) **HMI must:**Store the provided information associating it with application's appID.
- 2) Change the display language for the application which requested the one
- 3) Respond to the request.

Note:

The applicable to this RPC result codes are provided in section 7.13.3 Response.

Note:

The sequence diagrams describing the expected HMI behavior are provided in the section 7.13.4 Sequence Diagrams

7.13.2.2 Parameters

Param Name	Type	Mandatory	Additional	Description
appName	string	false	maxlength = 100	Request new app name registration Needs to be unique over all applications. May not be empty. May not start with a new line character. May not interfere with any name or synonym of any registered applications. Applications with the same name will be rejected. (SDL makes all the checks)
ngnMediaScreenAppName	string	false	maxlength = 100	Request new app short name registration
language	Common.Language	true		The language requested to be switched to. See Language.
appHMIType	Common.AppHMIType	false	minsize=1 maxsize=100 array="true"	Sent when app's requested-during-registration AppHMIType is changed to the different one due to Policies update. Contains the updated list of all allowed app's AppHMITypes.
appID	Integer	true		ID of the application that relates to this RPC.

7.13.2.3 Language

Element 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

Element Name	Value	Description
ZH-CN	15	Mandarin – China
ZH-TW	16	Mandarin – Taiwan
JA-JP	17	Japanese – Japan
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 Belgium (Flemish)
EL-GR	25	Greek
HU-HU	26	Hungarian
FI-FI	27	Finnish
SK-SK	28	Slovak

7.13.3 Response

Note:

There is a difference in message type for WebSocket and D-Bus connection (see the table below).

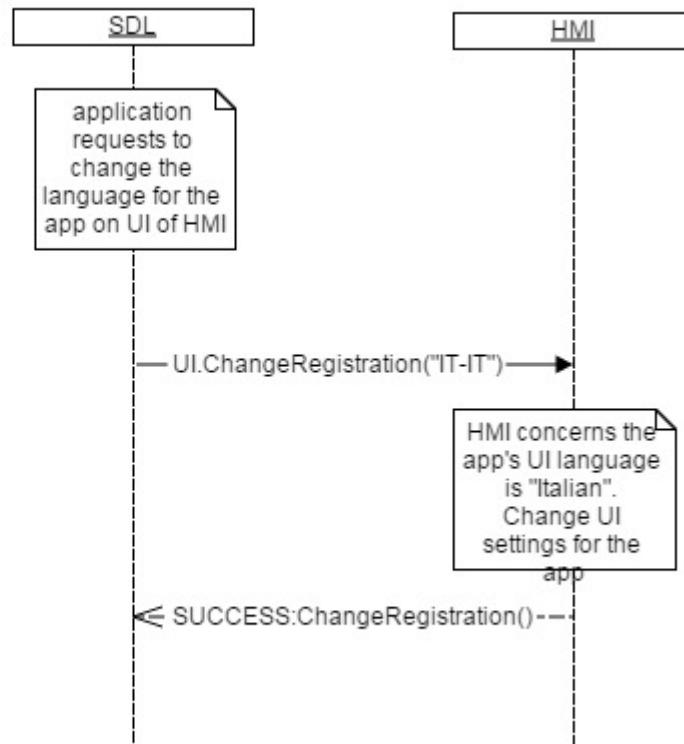
Result	Description	Message type		Mess age Params	Notes
		WebSo cket	D- Bus		
Success	SUCCESS: HMI has changed the UI language for the named application.	JSON respons e	Met hod return	code : 0	
Failure	INVALID_ID appID is not valid	JSON error messag e	Met hod return	code : 13	Applica ble for this RPC result codes. Please see Result Enumeration for all SDL-supported codes.
	WRONG_LAN GUAGE Language isn't supported by UI			code : 16	
	INVALID_D ATA: The data sent is invalid (invalid JSON)			code : 11 mess age	

	syntax or parameters out of bounds or of wrong type)			
	GENERIC_ERROR: 1) The unknown issue occurred or other codes are not applicable.		code : 22 message	

SDL 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.

7.14.4 Sequence Diagrams

7.14.4.1 ChangeRegistration



7.13.5 JSON Messages Examples

7.13.5.1 Request

{ "id" : 117,

```

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

```

7.13.5.2 Response

```

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

```

7.13.5.3 Error message

```

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

```

7.14 GetLanguage

7.14.1 Description

Type:	Function
Sender:	SDL
Purpose:	Find out the current HMI display language.

SDL inquires the current HMI display language.

Note:

'HMI display language' is meant to be the language of HMI-defined menus, screens, popups, etc.
HMI display language and application display language (requested via UI.ChangeRegistration) might be different.

This RPC is sent by SDL after UI readiness is confirmed by UI.IsReady. If a User changes the HMI display language subsequently, HMI must inform SDL about this event via UI.OnLanguageChange notification.

7.14.2 Request

7.14.2.1 Behavior

HMI must:

- 1) Check the HMI display language currently in effect. Respond providing SDL with the results of this check.

Note:

The applicable to this RPC result codes are provided in section 7.14.3 Response.

Note:

The sequence diagrams describing the expected HMI behavior are provided in the section 7.14.4 Sequence Diagrams

7.14.3 Response

Note:

There is a difference in message type for WebSocket and D-Bus connection (see the table below).

Result	Description	Message type		Message Params	Notes
		WebSocket	D-Bus		
Success	SUCCESS: The current HMI display language is provided.	JSON response	Method return	language, code: 0	
Failure	INVALID_DATA: The data sent is invalid (invalid JSON syntax)	JSON error message	Method return	code: 11	Applicable for this RPC result codes.
	DATA_NOT_AVAILABLE: The information of the current HMI display language cannot be provided.			Code: 9	
	GENERIC_ERROR: The unknown issue occurred or other codes are not applicable.			code: 22	Please see Result Enumeration for all SDL-supported codes.

7.14.3.1 Parameters

Param Name	Type	Mandatory	Description
------------	------	-----------	-------------

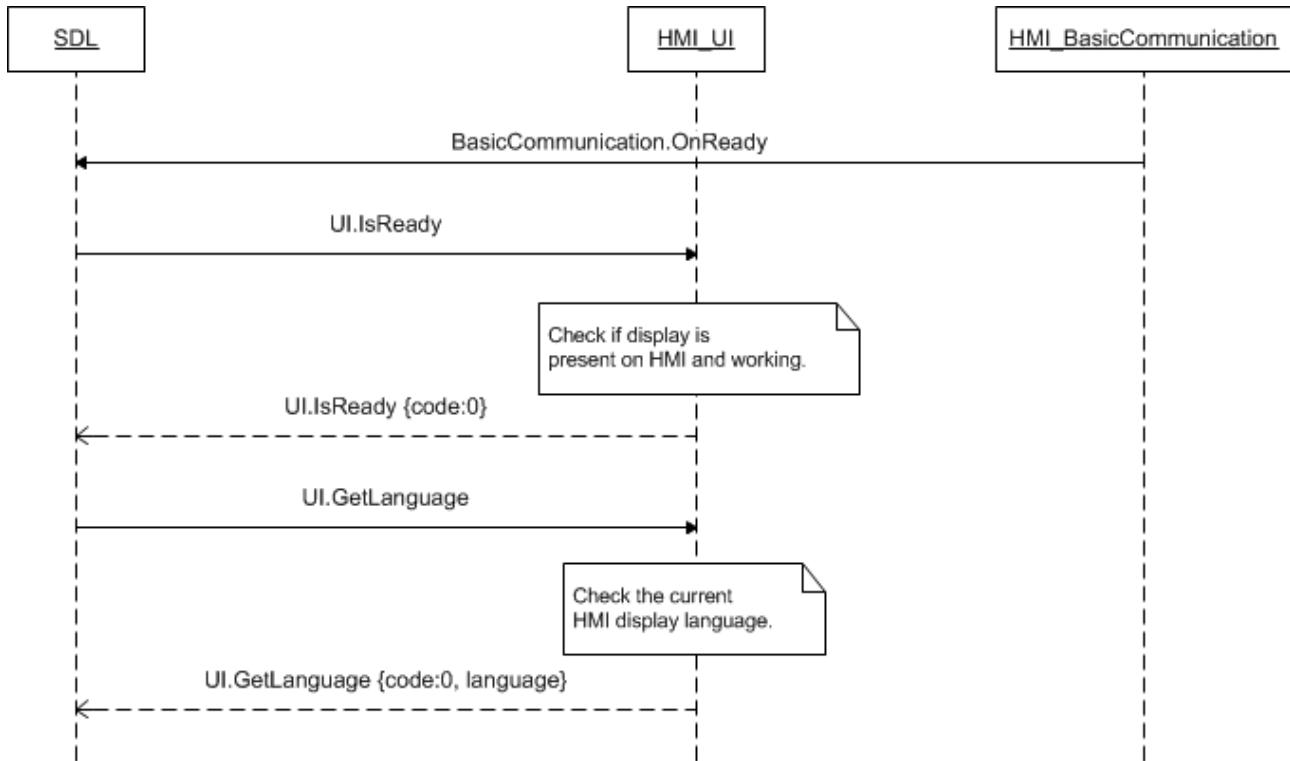
language	Common.Language	true	Display language that is currently active. See Language .
----------	-----------------	------	--

7.14.3.3 Language

Element 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
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 Belgium (Flemish)
EL-GR	25	Greek
HU-HU	26	Hungarian
FI-FI	27	Finnish
SK-SK	28	Slovak

7.14.4 Sequence Diagrams

7.14.4.1 GetLanguage



7.14.5 JSON Messages Examples

7.14.5.1 Request

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

7.14.5.2 Response

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

7.14.5.3 Error message

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

```

{
    "method" :
"UI.GetLanguage"
}
}
}

```

7.15 SetAppIcon

7.15.1 Description

Type:	Function
Sender:	SDL
Purpose:	Display the icon together with application name in the HMI application list

SDL requests to display the icon together with the specified application name (sent via OnAppRegistered or UpdateAppList to HMI) in the HMI list of registered applications. The path to the icon stored on HU is provided within request.

7.15.2 Request

7.15.2.1 Behavior

HMI must:

- 1) Store the provided data and associate it with application apID.
- 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) Respond the request.

HMI may:

Display the HMI default icons for those applications that did not send the SetAppIcon request yet.

Note:

- The applicable to this RPC result codes are provided in section 7.15.3 Response.
- The sequence diagrams describing the expected HMI behavior are provided in the section 7.15.4 Sequence Diagrams
- The picture of "Slider" screen is added to the section 7.15.5 Possible Layout.

7.15.2.2 Parameters

Param Name	Type	Mandatory	Description
syncFileName	Common.Image	true	- The path to the image stored on HU - Or the binary image itself. See Image.

Param Name	Type	Mandatory	Description
appID	Integer	true	ID of the application related to this RPC.

7.15.2.3 Image

Param 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
imageType	Common.ImageType	true	-	Describes, whether it is a static or dynamic image.

7.15.2.4 ImageType

Element name	Value	Short Description
STATIC	0	Static image. The image that is sent as just a hex code “syncFileName” value in the request. HMI identifies the code going through its predefined list of images stored on HMI.
DYNAMIC	1	Dynamic image. The image that is stored on HMI in a predefined shared folder (SystemFilesPath parameter in ini file, see 15.3 chapter)

7.15.3 Response

Note:

There is a difference in message type for WebSocket and D-Bus connection (see the table below).

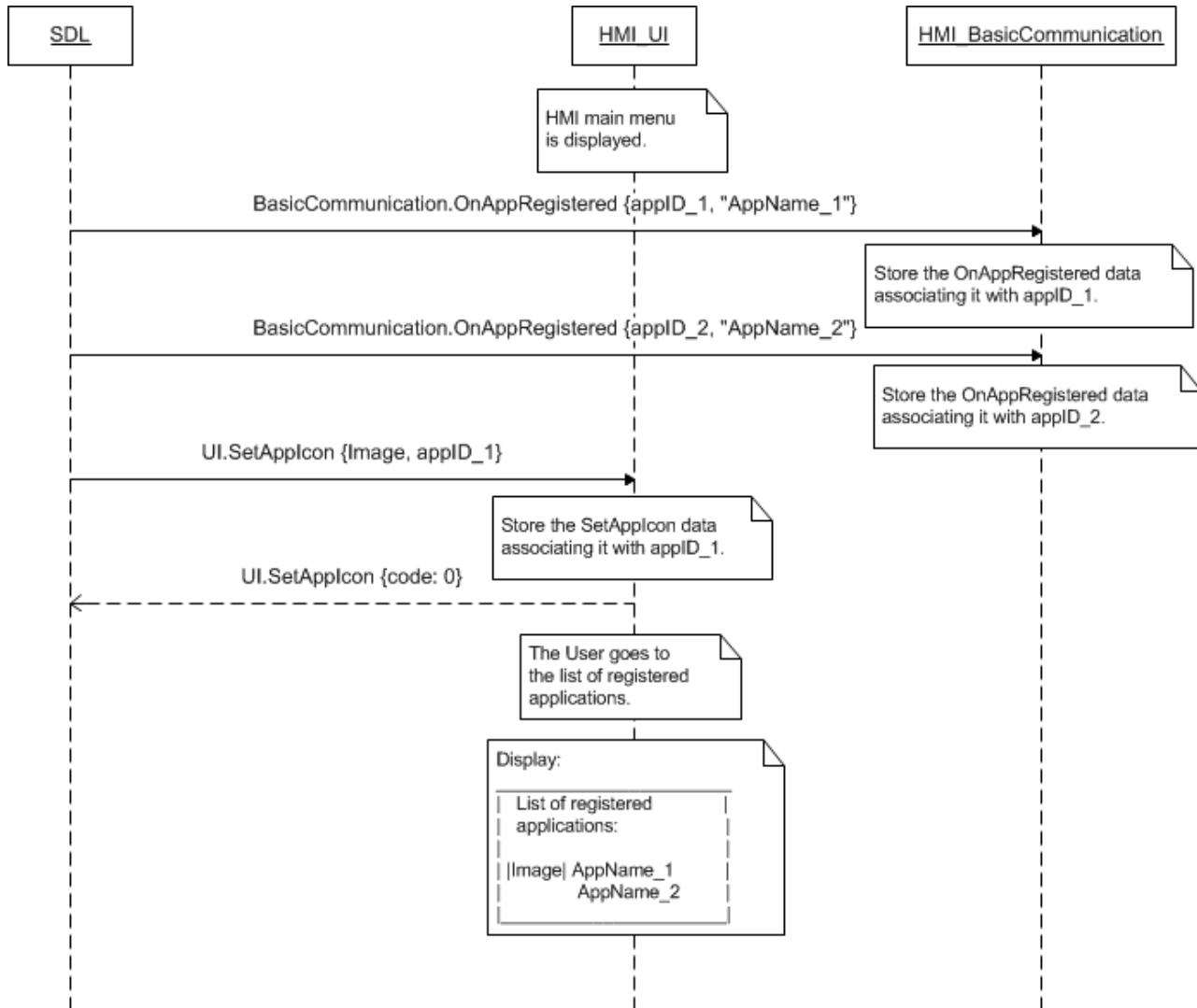
Result	Description	Message type		Message Params	Notes
		WebSocket	D-Bus		
Success	SUCCESS: The requested icon is displayed for the named application in the HMI list of registered applications.	JSON response	Method return	code : 0	
Failure	INVALID_DATA: The data sent is invalid (invalid JSON syntax or parameters out of bounds or of wrong type)	JSON error message	Method return	code : 11	Applicable for this RPC result codes. Please see Result Enumeration
	INVALID_ID appID is invalid (e.g. doesn't exist)			code : 13	

	<p>UNSUPPORTED_RESOURCE</p> <p>When icon is sent by SDL but HMI doesn't support the type of images sent by SDL (STATIC/DYNAMIC).</p>			code : 2	for all SDL-supported codes.
	<p>GENERIC_ERR</p> <p>OR:</p> <p>1) The unknown issue occurred or other codes are not applicable.</p>			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.

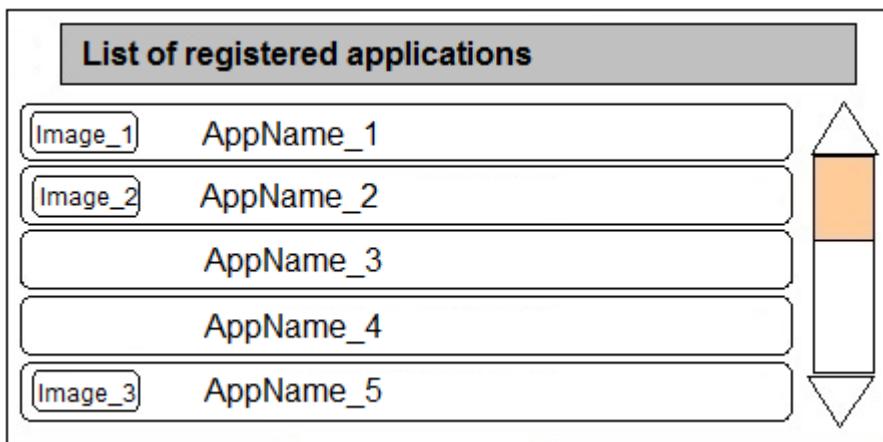
7.15.4 Sequence Diagrams

7.15.4.1 SetAppIcon



7.15.5 Possible Layout

7.15.5.1 The list of registered applications with and without icons



7.15.6 JSON Messages Examples

7.15.6.1 Request

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

7.15.6.2 Response

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

7.15.6.3 Error message

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

7.16 Slider

7.16.1 Description

Type:	Function
Sender:	SDL
Purpose:	Display the user-controlled slider.

SDL requests to display the user-controlled full screen or popup slider (see the picture in section 7.16.5 Possible Layout).

The request arrives only for the application currently active on HMI.

7.16.2 Request

7.16.2.1 Behavior

HMI must:

1. Display the user-controlled slider (see the picture in section 7.16.5 Possible Layout) with:
 - The number of ticks defined by `nnumTicks` parameter
 - The initial position of the slider control corresponding to `position` parameter
 - The header defined by `sliderHeader` parameter
 - The footer defined by `sliderFooter` parameter:
 - Static is a single text string that must be displayed throughout any positions chosen.
 - Dynamic is an array of strings that must be assigned for every position and displayed when position is set-up by the User.
2. Keep displaying slider until one of the following events occurs:
 - Timeout defined by `timeout` parameter
 - The slider is aborted by the User or any RPC of the higher priority
 - The User presses HMI-defined ‘OK’ button.
3. Respond the request (providing `position` parameter in case the slider is closed by ‘OK’ button press or aborted, see section 7.16.3 Response).

HMI may:

Display the HMI-defined ‘Close’/‘Back’ button by the press of which the slider popup/screen must be closed (HMI must not provide any notifications to SDL upon such button press).

Note:

- The applicable to this RPC result codes are provided in section 7.16.3 Response.
- The sequence diagrams describing the expected HMI behavior are provided in the section 7.16.4 Sequence Diagrams
- The picture of “Slider” screen is added to the section 7.16.5 Possible Layout.

7.16.2.2 Parameters

Param Name	Type	Mandatory	Additional	Description
<code>numTicks</code>	Integer	true	<code>minvalue = 2</code> <code>maxvalue = 26</code>	Number of selectable items on a horizontal axis to be displayed.
<code>position</code>	Integer	true	<code>minvalue = 1</code> <code>maxvalue = 26</code>	Initial position of the slider control.

Param Name	Type	Mandatory	Additional	Description
sliderHeader	String	true	maxlength = 500	Text header of the slider to be displayed.
sliderFooter	String	false	Array = true maxlength = 500 minsize = 1 maxsize = 26	Text footer to be displayed. For the <i>static</i> text footer, the only footer string is provided in the array. For a <i>dynamic</i> text footer: - The number of footer text strings in the array always matches the numTicks value. - Text array string should correlate with potential slider position index. If omitted, no footer text must be displayed.
timeout	Integer	true	minvalue = 1000 maxvalue = 65535	The timeout in milliseconds. It is the amount of time the HMI must wait for User's action while displaying the slider. When the amount of time defined has elapsed, the HMI must close the dialog with slider and return to the layout previously displayed.
appID	Integer	true	-	ID of the application related to this RPC.

7.16.3 Response

Note:

There is a difference in message type for WebSocket and D-Bus connection (see the table below).

Result	Description	Message type		Message Params	Notes
		WebSocket	D-Bus		
Success	SUCCESS: HMI displayed the slider with requested parameters and has just closed it by 'OK' button press. The last slider position must be returned.	JSON response	Method return	sliderPosition, code: 0	
Failure	ABORTED: HMI has displayed the slider with requested parameters and has just closed it by: 1) HMI-defined 'Return'/Back' button press. 2) A higher priority RPC. The last slider position must be returned.	JSON error message	Method return	sliderPosition, code: 5	Applicable for this RPC result codes. Please see Result Enumeration for all SDL-supported codes.
	TIMED_OUT: HMI displayed the slider with requested parameters and has just closed it by timeout.			code: 10	

	INVALID_DATA: The data sent is invalid (invalid JSON syntax or parameters out of bounds or of wrong type)		code: 11	
	INVALID_ID appID is invalid (e.g. doesn't exist)		code: 13	
	GENERIC_ERROR: 1) The unknown issue occurred or other codes are not applicable.		code: 22	

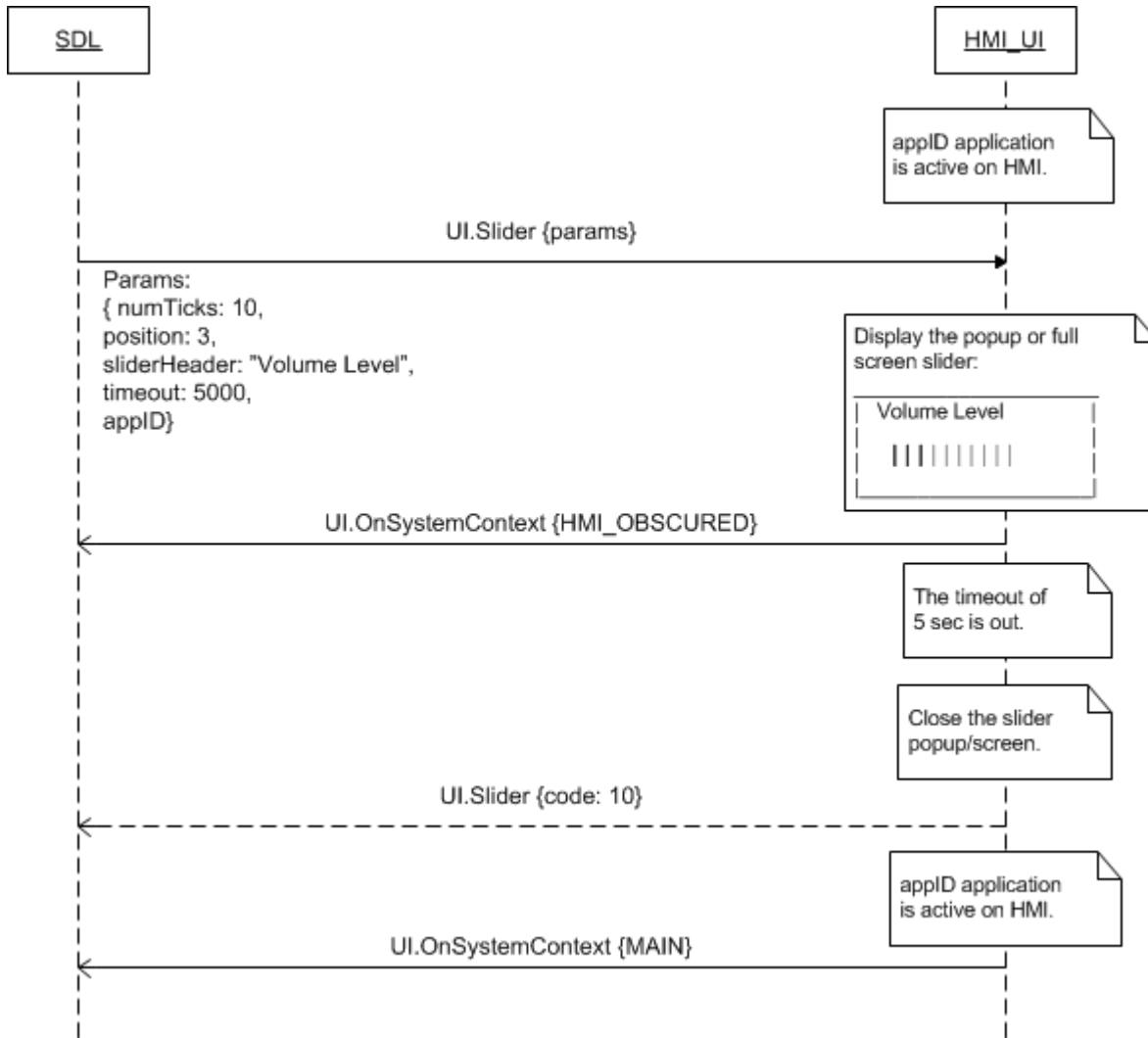
SDL Note: In case HMI does not respond SDL's request during request/SDL defined timeout, SDL will return GENERIC_ERROR result code to the corresponding mobile app's request.

7.16.3.1 Parameters

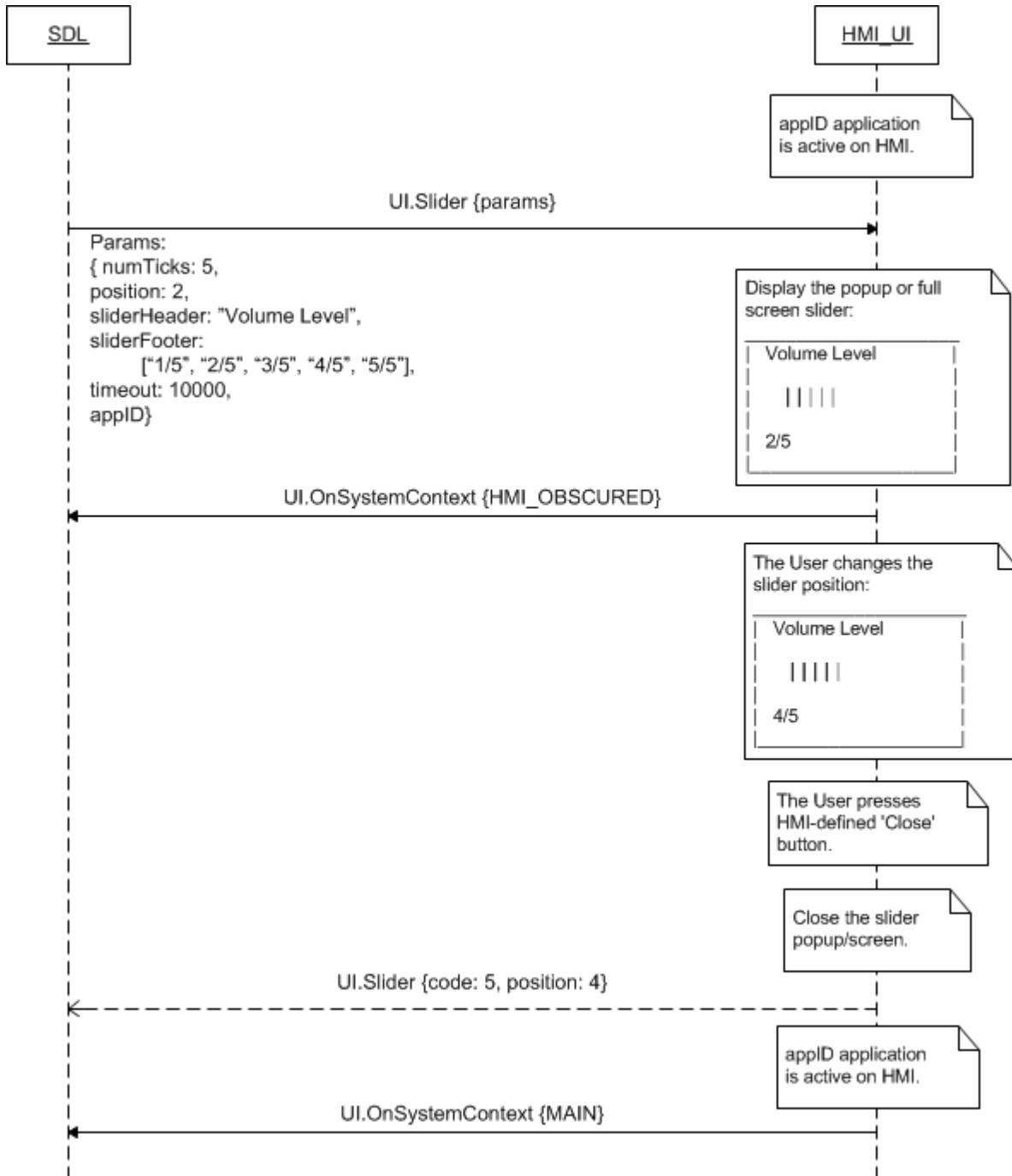
Param Name	Type	Mandatory	Additional	Description
sliderPosition	Integer	true	minvalue = 1 maxvalue = 26	Current slider value to be returned when the slider is closed by the timeout or aborted.

7.16.4 Sequence Diagrams

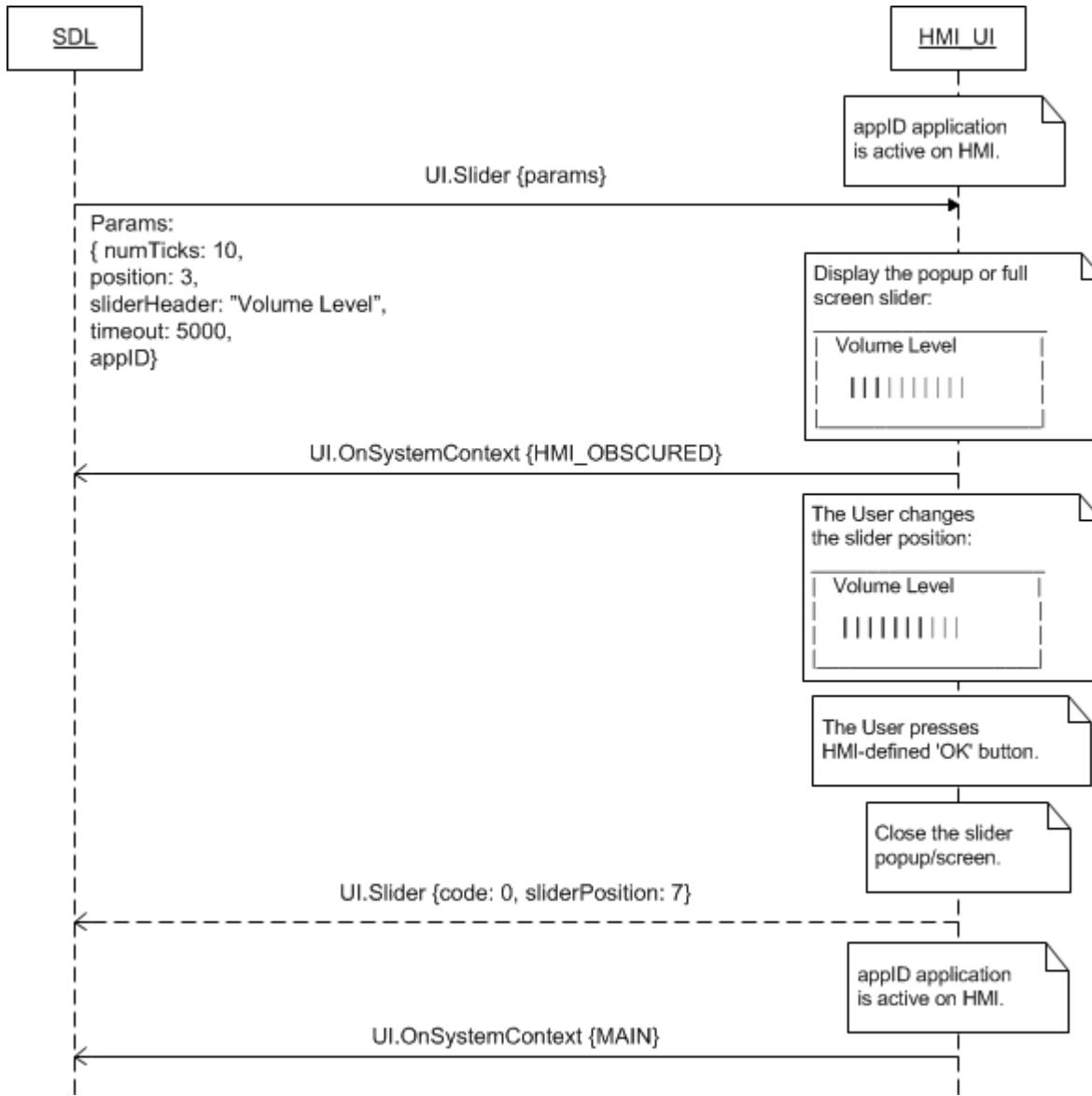
7.16.4.1 Slider with static footer displayed then closed by the timeout



7.16.4.2 Slider with dynamic footer displayed then aborted

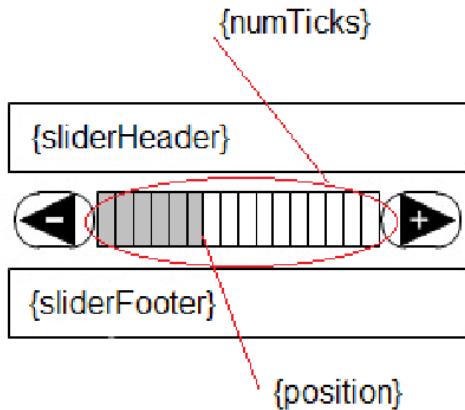


7.16.4.2 Slider with static footer displayed then closed by 'OK' button press

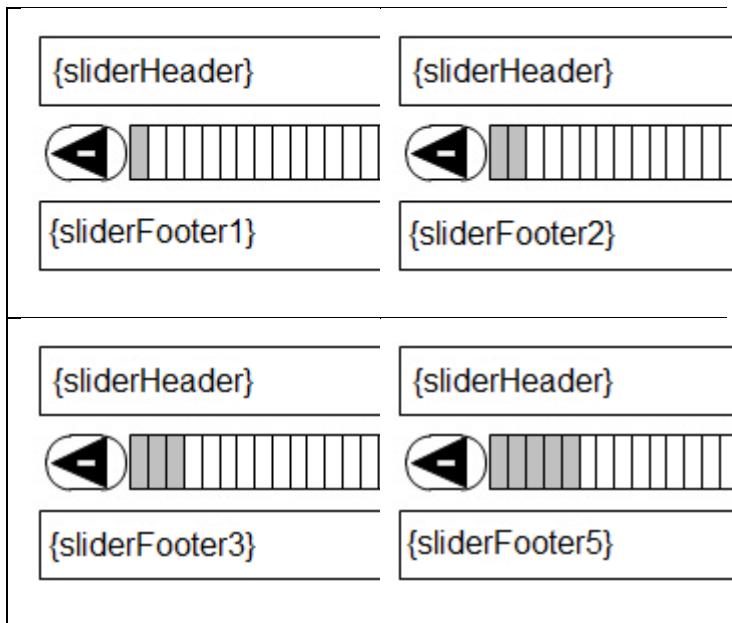


7.16.5 Possible Layout

7.16.5.1 Slider



7.16.5.2 Slider with dynamic footer



7.16.6 JSON Messages Examples

7.16.6.1 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  
    }  
}
```

7.16.6.2 Response

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

7.16.6.3 Error message

```
{  
    "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"  
        }  
    }  
}
```

7.17 ScrollableMessage

7.17.1 Description

Type:	Function
Sender:	SDL
Purpose:	Display the dialog that may contain new lines and tabs (for long texts displaying).

SDL requests to display a popup or full screen overlay
with the text that might be formatted with the tabs and new
lines.

The request may arrive in both cases when the application
is 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).

7.17.2 Request

7.17.2.1 Behavior

HMI must:

1. Display the popup or full screen overlay of formatted
text together with up to 8 soft buttons (or without any,
depending on parameters received).

Important Note:

HMI must provide `OnButtonPress`/`OnButtonEvent` notifications for every soft button defined within request in case the application subscribed on `CUSTOM_BUTTON` via [OnButtonSubscription](#) notification.

2. Keep displaying the message until:

- The time is out defined by the timeout parameter

Important Note:

HMI must renew the timeout informing SDL about this event via `UI.OnResetTimeout` notification after

- a) Every User's action/touch over the message (e.g. the User scrolls the message)
 - b) `KEEP_CONTEXT` soft button (defined within this RPC) press.
-
- One of the events below occurs:
 - HMI-defined 'Close' button press
 - ClosePopup RPC from SDL
 - RPC or system event of a higher priority
 - Soft button of `DEFAULT_ACTION` or `STEAL_FOCUS` press (if the soft button of such system action is defined within `ScrollableMessage` request)

3. Respond the request.

Note:

- The applicable to this RPC result codes are provided in section 7.17.3 Response.
- The sequence diagrams describing the expected HMI behavior are provided in the section 7.17.4 Sequence Diagrams
- The picture of scrollable message screen is added to the section 7.17.5 Possible Layout.

7.17.2.2 Parameters

Param Name	Type	Mandatory	Additional	Description
messageText	Common.TextFieldStruct	true	–	The body of text that can include newlines and tabs and can be scrolled up. See <code>TextFieldStruct</code> . Uses <code>scrollableMessageBody</code> (see <code>TextFieldName</code>).

timeout	Integer	true	Minvalue = 1000 maxvalue = 65535	The timeout in milliseconds defined by the application. It is the amount of time that - HMI must keep displaying the message - Must be renewed after every User's action/touch (e.g. scrolling the message) with sending UI.OnResetTimeout notification to SDL. When the defined amount of time has elapsed, HMI must - close the message popup/screen - send the successful response.
softButtons	Common.SoftButton	false	Array = true Minsize = 0 Maxsize = 8	Application defined soft buttons (See SoftButton). HMI must provide OnButtonPress/OnButtonEvent notifications for every soft button defined within request. If omitted HMI may display the system-defined "Close" soft button providing the response with ABORTED Result code when it is pressed. HMI must not send any notifications to SDL for such button.
appID	Integer	true	-	ID of the application that concerns this RPC.

7.17.2.3 TextFieldStruct Structure

Param Name	Type	Mandatory	Additional	Description
fieldName	Common.TextFileDialogName	true	-	The name of the field where the text must be displayed in.
fieldText	String	true	Maxlength = 500	The text to be displayed.

7.17.2.4 TextFieldName Enumeration

Only the text fields applicable to Alert RPC are described within this section. All the text fields names recognized by SDL are described in the section [14.1.15 TextFieldName](#).

Element name	Short Description
scrollableMessageBody	The long form body of text that can include newlines and tabs. Applies to ScrollableMessage, section 7.17 .

7.17.3 Response

Note:

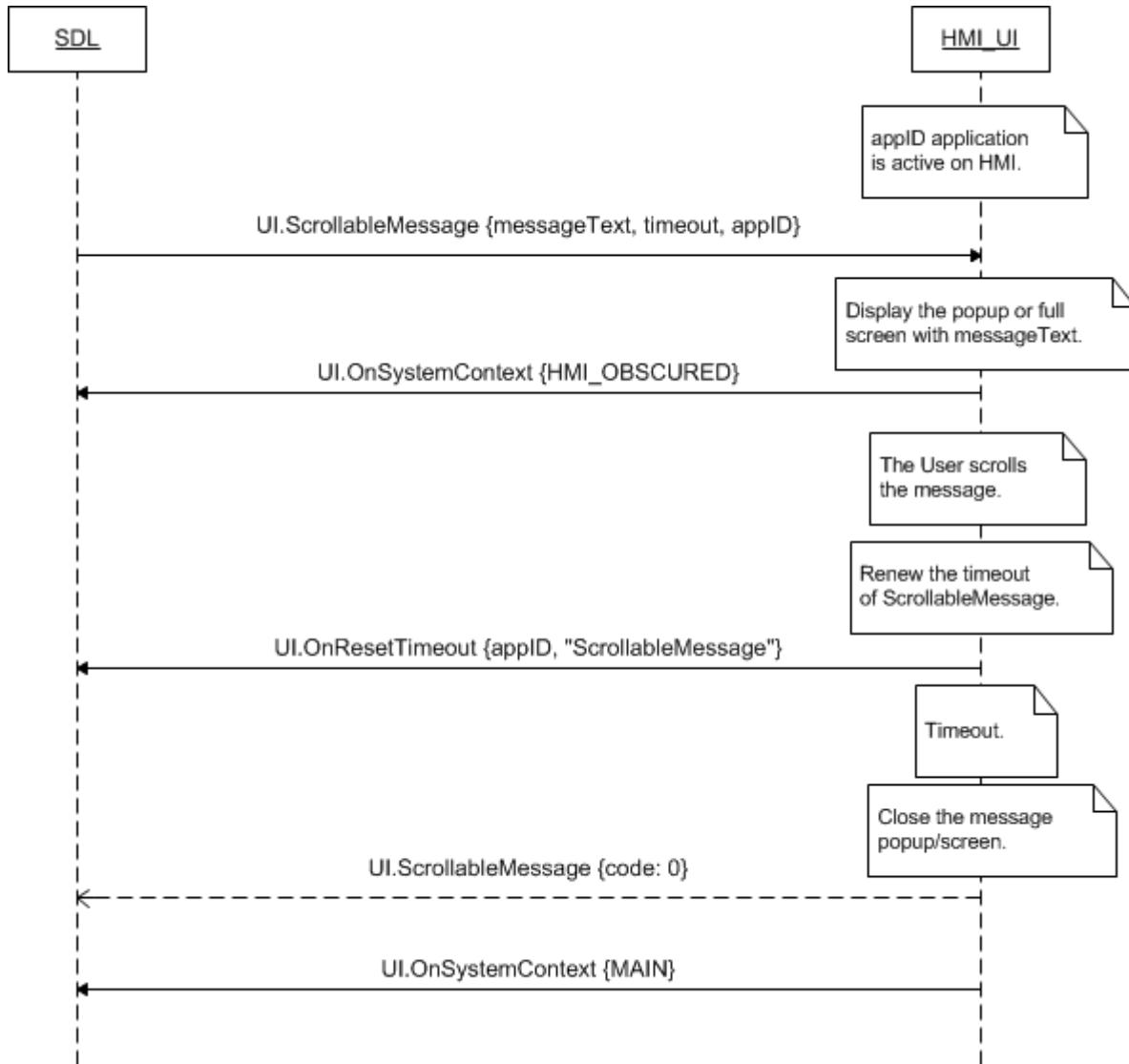
There is a difference in message type for WebSocket and D-Bus connection (see the table below).

Result	Description	Message type		Message Params	Notes
		WebSocket	D-Bus		
Success	SUCCESS: HMI displayed the message with requested parameters and has just closed it by 1) The timeout. 2) STEAL_FOCUS soft button (defined within this RPC) press.	JSON response	Method return	code: 0	
Failure	ABORTED: HMI displayed the message with requested parameters and has just closed it by: 1) HMI-defined 'Close'/'Back' button press. 2) A higher priority RPC or system event 3) DEFAULT_ACTION soft button (defined within this RPC) press.	JSON error message	Method return	code: 5	Applicable for this RPC result codes. Please see Result Enumeration for all SDL-supported codes.
	CHAR_LIMIT_EXCEEDED There's char limit exceeded for the whole text of Slider message on the screen			code: 12	
	INVALID_DATA: The data sent is invalid (invalid JSON syntax or parameters out of bounds or of wrong type)			code: 11	
	INVALID_ID appID is invalid (e.g. doesn't exist)			code: 13	
	GENERIC_ERROR: 1) The unknown issue occurred or other codes are not applicable.			code: 22	

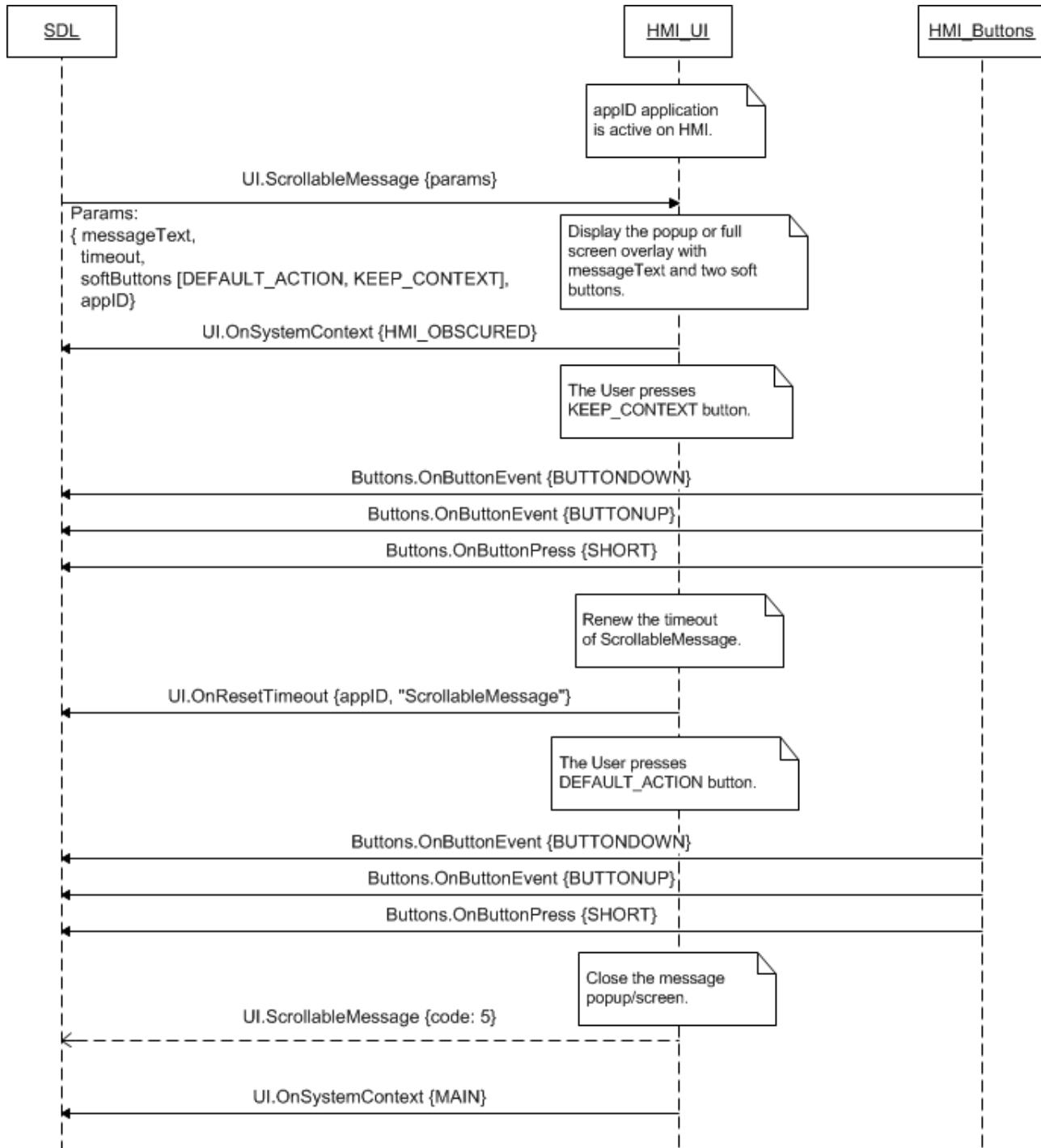
SDL Note: In case HMI does not respond SDL's request during SDL/request defined timeout, SDL will return GENERIC_ERROR result code to the corresponding mobile app's request.

7.17.4 Sequence Diagrams

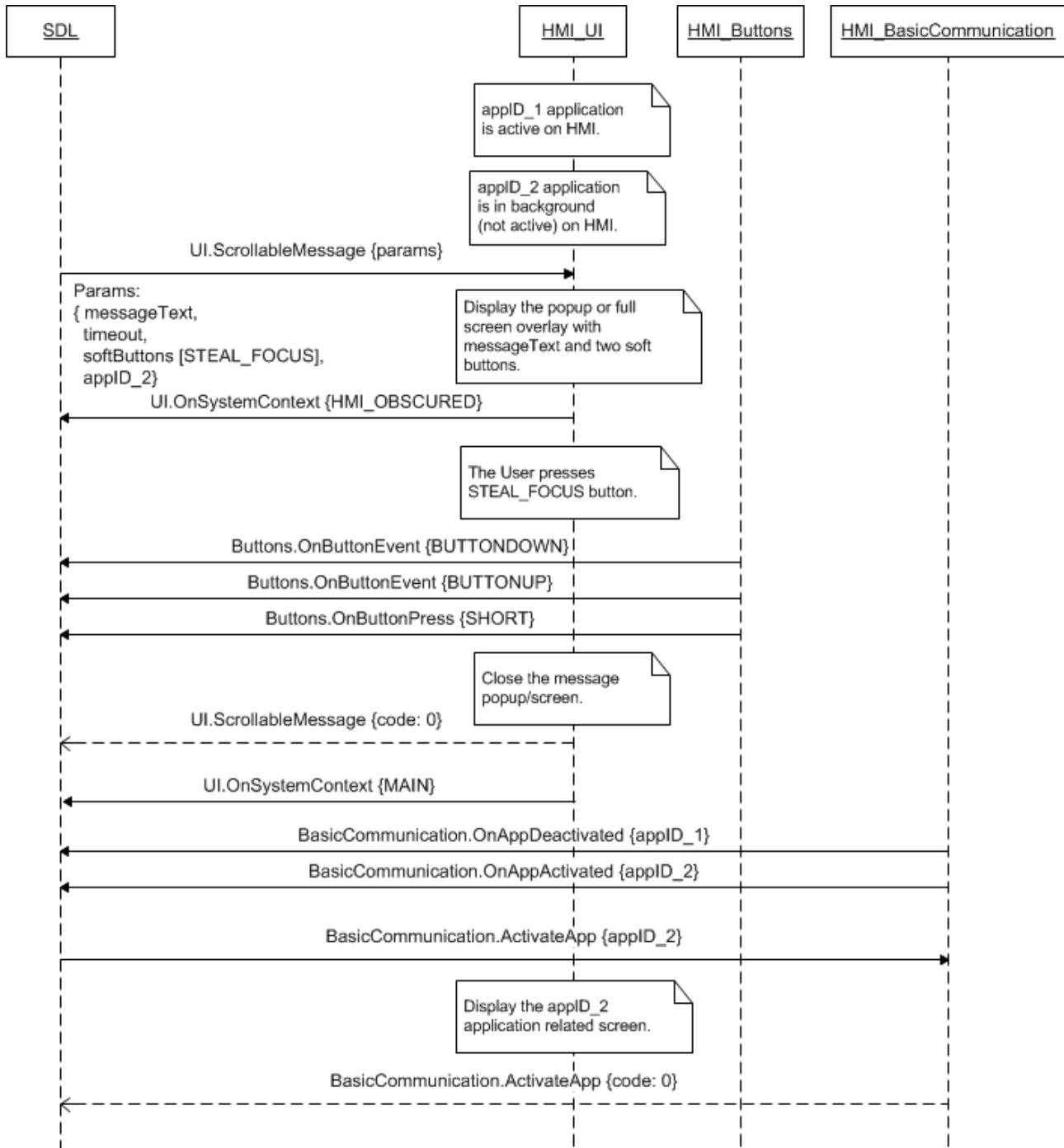
7.17.4.1 ScrollableMessage displayed, scrolled by the User and closed by the timeout



7.17.4.2 ScrollableMessage with soft buttons of `DEFAULT_ACTION` and `KEEP_CONTEXT` system action pressed by the User



7.17.4.3 *ScrollableMessage with STEAL_FOCUS soft button for the application not active on HMI*



7.17.5 JSON Messages Examples

7.17.5.1 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
}
}

```

7.17.5.2 Response

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

7.17.5.3 Error message

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

7.18 PerformAudioPassThru

7.18.1 Description

Type:	Function
Sender:	SDL
Purpose:	Start audio capturing.

SDL prompts HMI to start audio capturing and inform the User about the event by displaying the dialog with the requested text information.

The request may arrive in both cases of active and background application on HMI.

Note:

This RPC may arrive together with TTS.Speak which purpose is to inform the User about start of audio capturing by the means of TTS module.

7.18.2 Request

7.18.2.1 Behavior

HMI must:

- 1) Display the dialog with requested text.
(audioPassThruDisplayText1 and
audioPassThruDisplayText2 text fields).
- 2) Attenuate or mute any audio source except of
TTS speaking (depending on HMI capabilities).
- 3) Start the audio capturing from the microphone.

Important Note:

If PerformAudioPassThru request is accompanied with TTS.Speak RPC, HMI must start the audio capturing only after TTS finishes speaking the requested text.

- 4) Keep displaying the dialog and capturing the audio until:

- The value of `maxDuration` is reached.
 The User presses any of HMI-defined
 'Cancel'/'Done'/'Retry' buttons
 The request of `EndAudioPassThru` comes from
 SDL.
 5) Respond the request.

Important: SDL performs audio data capturing and transferring to mobile side by itself (no HMI API are involved). There're two options:

a) SDL is built with “-
`DEXTENDED_MEDIA_MODE=OFF`” flag (for the testing purposes). In this case SDL reads the hardcoded .waw file (filename is *RecordingFileSource* located in *AppStorageFolder*, see [15.1.2.2 MAIN](#)) and sends data read via `OnAudioPassThru` to mobile application. For more details refer [the diagram 7.18.4.3](#).

b) SDL is built with “-
`DEXTENDED_MEDIA_MODE=ON`”. In this case SDL reads the data from the microphone connected to PC and sends data read via `OnAudioPassThru` to mobile application. For more details see [the diagram 7.18.4.4](#).

HMI may:

Display the HMI-defined 'Done', 'Retry', 'Cancel' soft buttons. HMI must not return the `OnButtonEvent`/`OnButtonPress` notifications to SDL when such soft button is pressed by the User.

Note:

- The applicable to this RPC result codes are provided in section 7.18.3 Response.
- The sequence diagrams describing the expected HMI behavior are provided in the section 7.18.4 Sequence Diagrams

7.18.2.2 Parameters

Param Name	Type	Mandatory	Additional	Description
<code>audioPassThruDisplayTexts</code>	<code>Common.TextFieldStruct</code>	true	<code>Array = true</code> <code>minsize = 0</code> <code>maxsize = 2</code>	The text to be displayed upon the start of audio capturing. See <code>TextFieldStruct</code> . Uses <code>audioPassThruDisplayText1</code> and <code>audioPassThruDisplayText2</code> (See <code>TextFieldName</code>)
<code>maxDuration</code>	<code>Integer</code>	true	<code>Minvalue = 1</code> <code>Maxvalue = 1000000</code>	The maximum duration of audio recording in milliseconds
<code>muteAudio</code>	<code>Boolean</code>	true	–	Defines if the current audio source should be muted during the APT session. If not, the audio source will play without interruption. If omitted, the value is set to true

appID	Integer	true	-	ID of the application that requested this RPC.
-------	---------	------	---	--

7.18.2.3 TextFieldStruct Structure

Param Name	Type	Mandatory	Additional	Description
fieldName	Common.TextFileldName	true	-	The name of the field where the text must be displayed in.
fieldText	String	true	Maxlength = 500	The text to be displayed.

7.18.2.4 TextFieldName Enumeration

Only the text fields applicable to PerformAudioPassThru RPC are described in this subsection. All the text fields names recognized by SDL are described in the section [14.1.15 TextFieldName](#).

Element name	Short Description
audioPassThruDisplayText1	The first line of text that must be displayed during audio capture.
audioPassThruDisplayText2	The second line of text that must be displayed during audio capture.

7.18.3 Response

Note:

There is a difference in message type for WebSocket and D-Bus connection (see the table below).

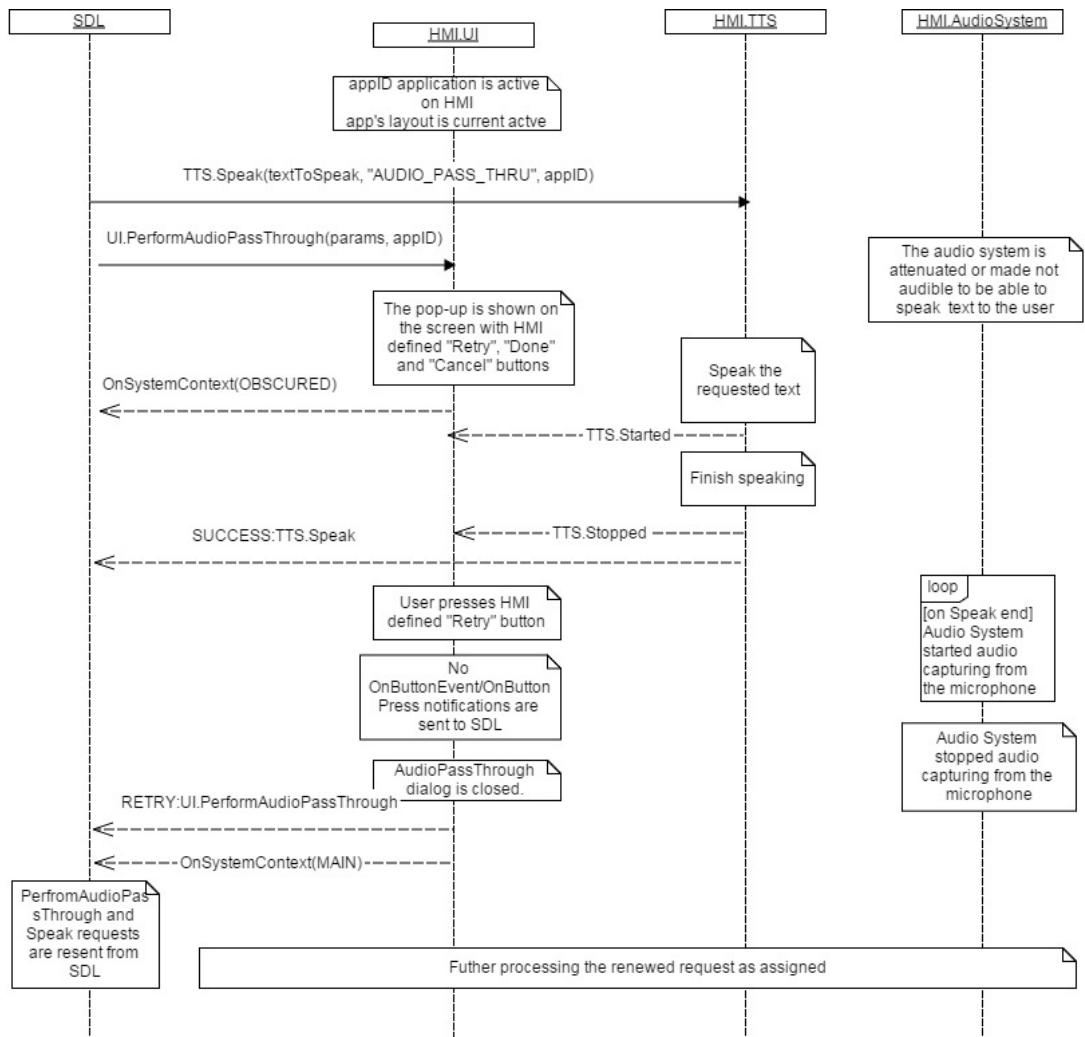
Result	Description	Message type		Message Params	Notes
		WebSocket	D-Bus		
Success	SUCCESS: HMI displayed the PerformAudioPassTru dialog with requested parameters, has just closed it and stopped the audio capturing by: 1) The value of <code>maxDuration</code> reached 2) The HMI-defined 'Ok'/'Done' button press 3) EndAudioPassTru request from SDL.	JSON response	Method return	code: 0	
Failure	ABORTED: HMI displayed the PerformAudioPassTru dialog with requested parameters, has just closed it and stopped the audio capturing by HMI-defined 'Cancel' button press.	JSON error message	Method return	code: 5	Applicable for this RPC result codes. Please see Result Enumeration for all SDL-supported

	<p>RETRY: HMI displayed the PerformAudioPassTru dialog with requested parameters, has just closed it and stopped the audio capturing by HMI-defined 'Retry' button press.</p> <p>INVALIDDATA: The data sent is invalid (invalid JSON syntax or parameters out of bounds or of wrong type)</p> <p>INVALIDID appID is invalid (e.g. doesn't exist)</p> <p>GENERICERROR: 1) The unknown issue occurred or other codes are not applicable.</p>		Code: 7	codes.
			code: 11	
			code: 13	
			code: 22	

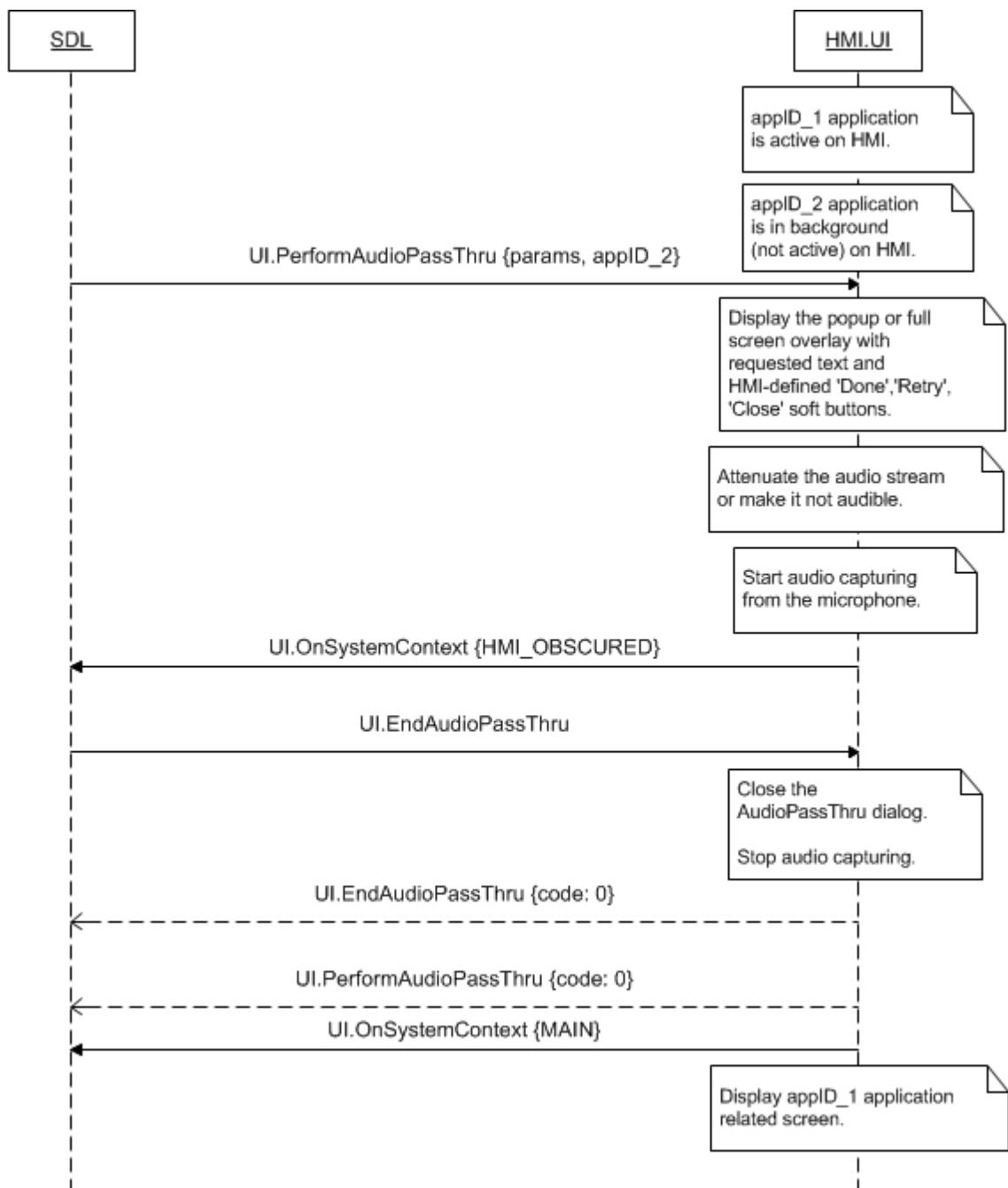
SDL Note: In case HMI does not respond SDL's request during max defined duration, SDL will return GENERIC_ERROR result code to the corresponding mobile app's request.

7.18.4 Sequence Diagrams

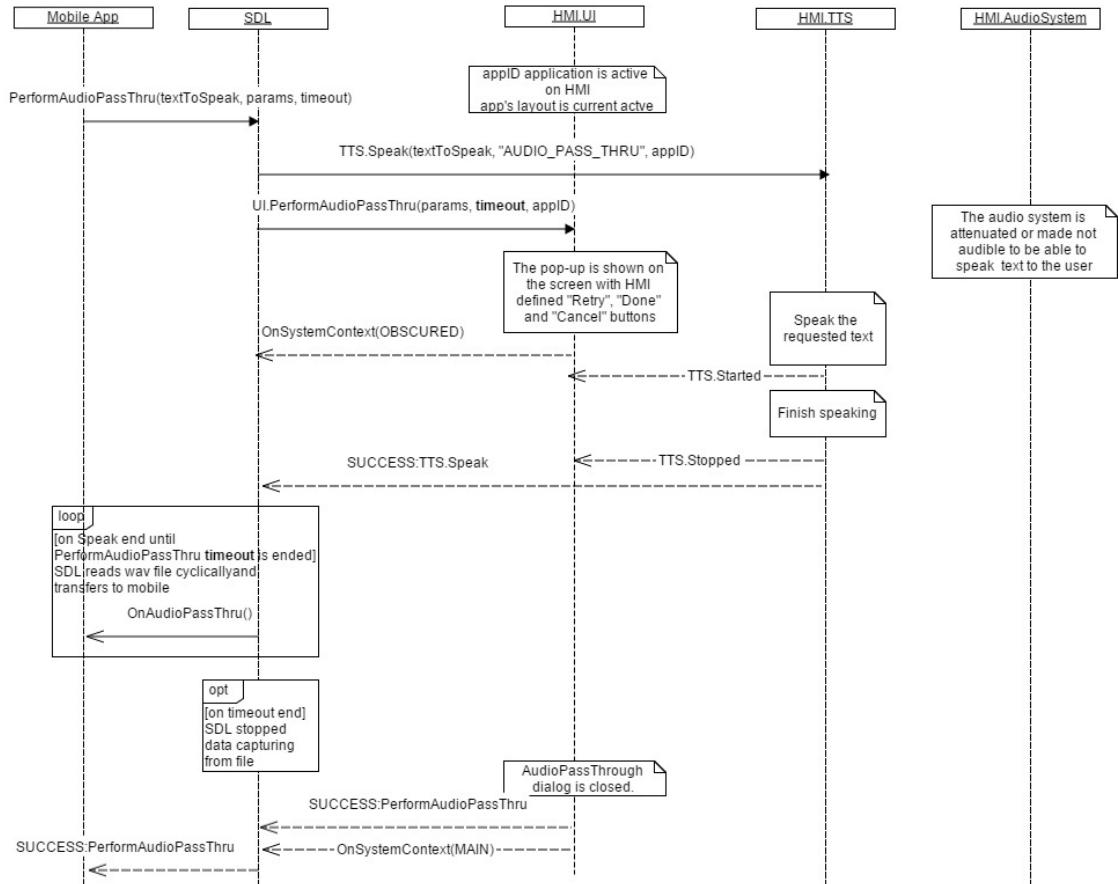
**7.18.4.1 PerformAudioPassTru requested
together with TTS.Speak, processed and then
finished by 'Retry' soft button press**



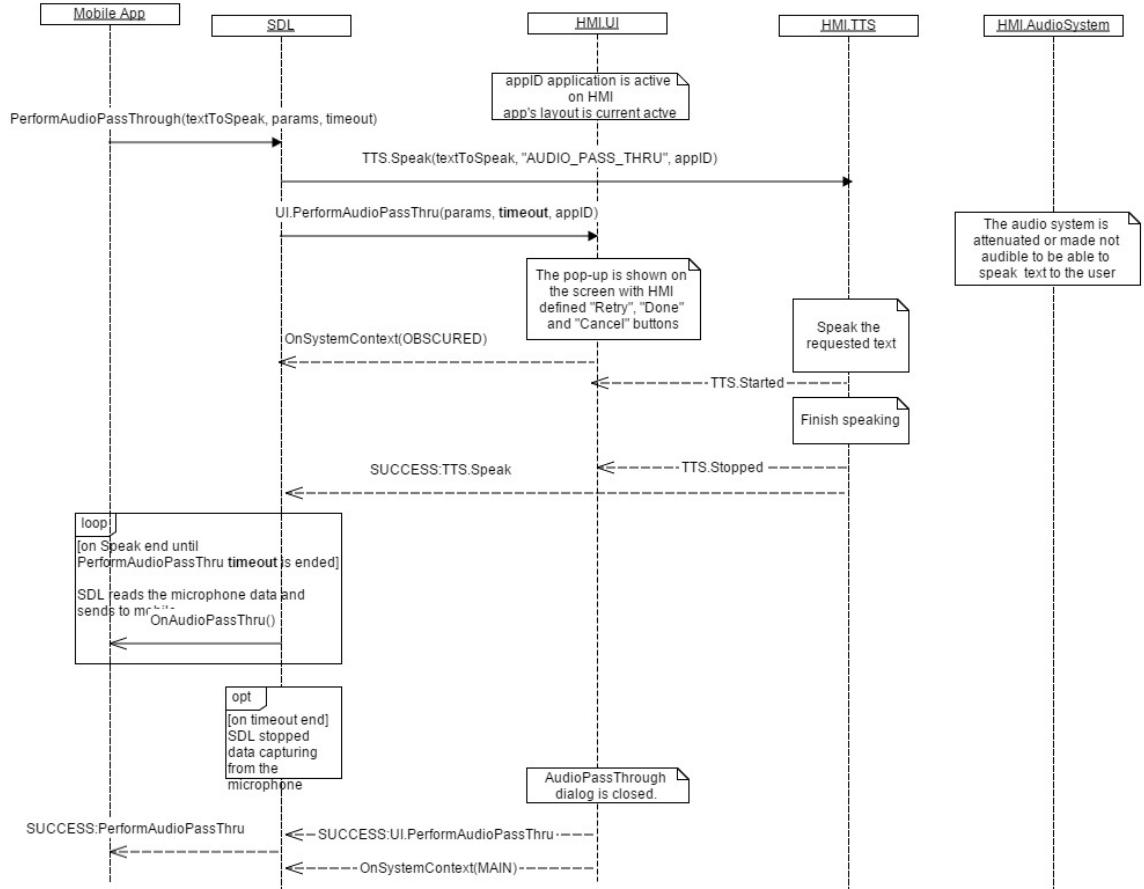
7.18.4.2 PerformAudioPassTru for the application not active on HMI, processed and then finished by EndAudioPassThru



7.18.4.3 PerformAudioPassThru for the application active on HMI, SDL built with with “-DEXTEDDED_MEDIA_MODE=OFF” flag (wav file reading)



7.18.4.4 PerformAudioPassThru for the application active on HMI, SDL built with with “-DEXTED_MEDIA_MODE=ON” flag (reading from the microphone)



7.18.5 JSON Messages Examples

7.18.5.1 Request

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

7.18.5.2 Response

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

7.18.5.3 Error message

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

7.19 EndAudioPassThru

7.19.1 Description

Type:	Function
Sender:	SDL
Purpose:	Stop audio capturing.

SDL prompts HMI to stop audio capturing and close the AudioPassThru dialog on UI.

7.19.2 Request

7.19.2.1 Behavior

HMI must:

- 1) Stop audio capturing started by PerformAudioPassThru RPC.
- 2) Close the PerformAudioPassThru dialog.
- 3) Respond the request.

Note:

HMI must first respond the EndAudioPassThru request, and then provide the response to the corresponding PerformAudioPassThru request.

Note:

- The applicable to this RPC result codes are provided in section 7.19.3 Response.

- The sequence diagrams describing the expected HMI behavior are provided in the section 7.19.4 Sequence Diagrams

7.19.3 Response

Note:

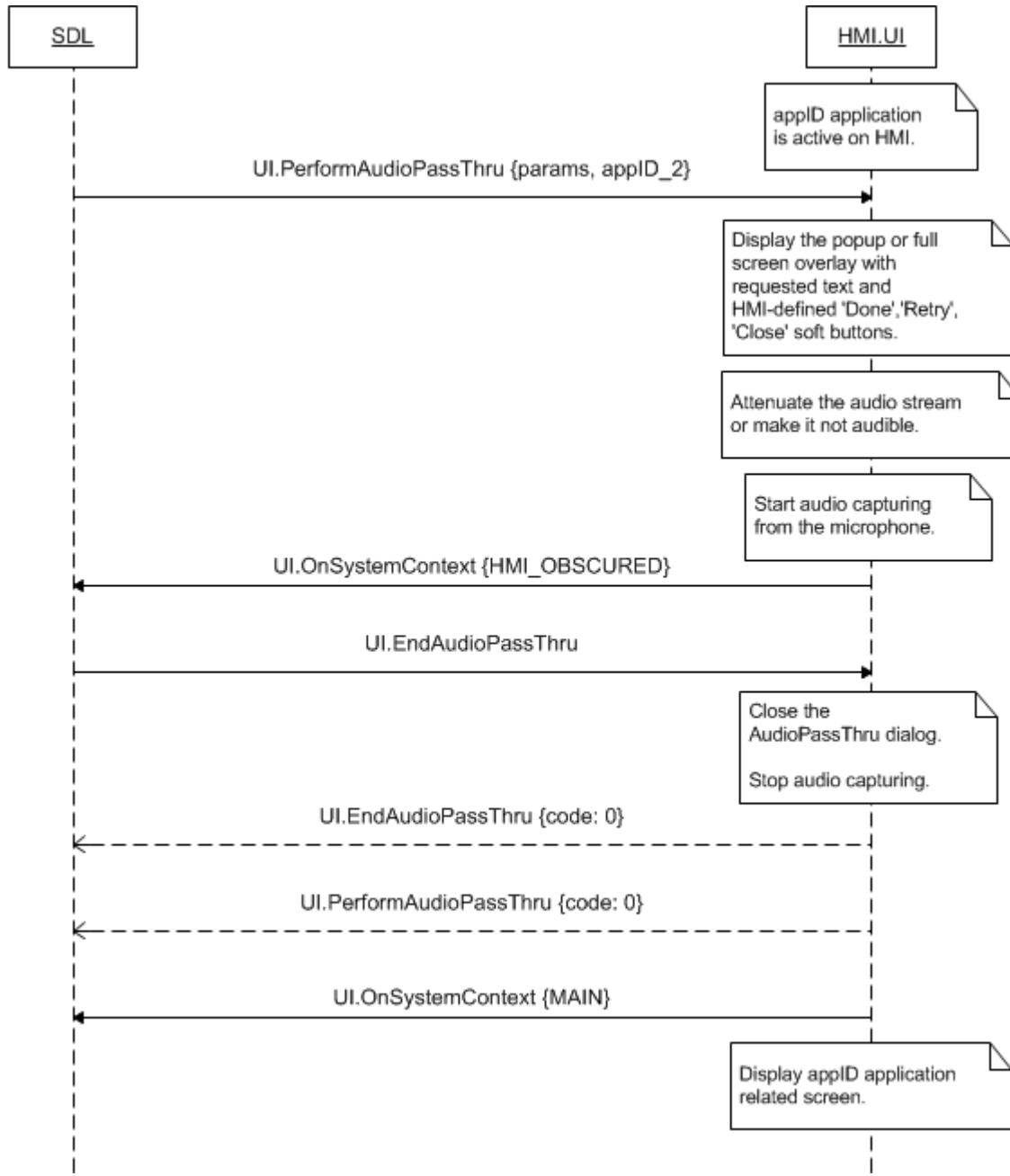
There is a difference in message type for WebSocket and D-Bus connection (see the table below).

Result	Description	Message type		Message Params	Notes
		WebSocket	D-Bus		
Success	SUCCESS: HMI has stopped audio capturing upon this request receipt.	JSON response	Method return	code: 0	
Failure	REJECTED: PerformAudioPassThru has already finished by the time this request arrived.	JSON error message	Method return	code: 4	Applicable for this RPC result codes. Please see Result Enumeration for all SDL-supported codes.
	INVALID_DATA: The data sent is invalid (invalid JSON syntax or parameters out of bounds or of wrong type)			code: 11	
	GENERIC_ERROR 1) The unknown issue occurred or other codes are not applicable.			code: 22	

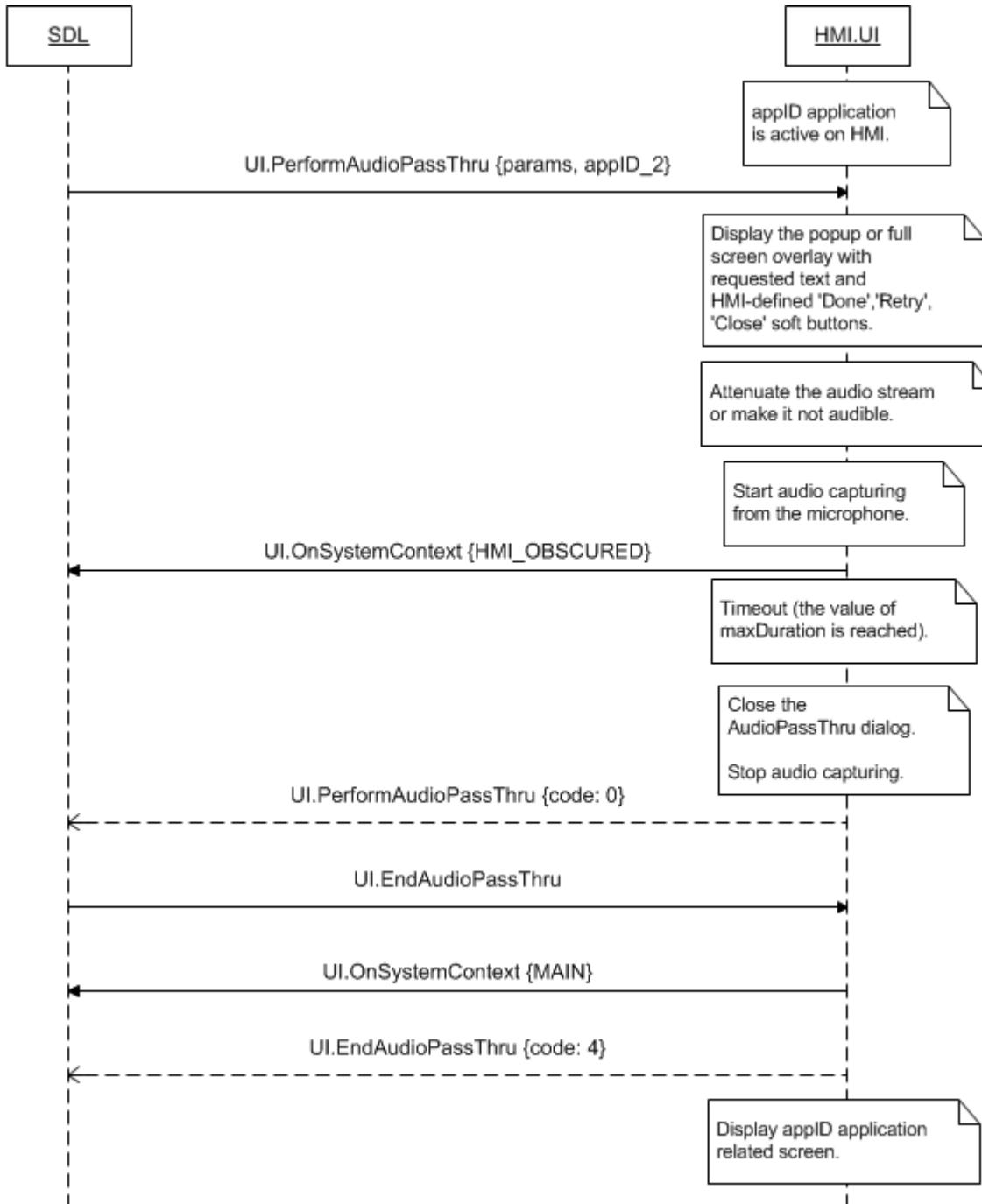
SDL Note: In case HMI does not respond SDL's request during SDL default timeout (10 sec), SDL will return GENERIC_ERROR result code to the corresponding mobile app's request.

7.19.4 Sequence Diagrams

7.19.4.1 EndAudioPassThru that stops the audio capturing



7.19.4.2 EndAudioPassThru for the case when audio capturing is already over



7.19.5 JSON Messages Examples

7.19.5.1 Request

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

7.19.5.2 Response

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

7.19.5.3 Error message

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

7.20 ClosePopUp

7.20.1 Description

Type:	Function
Sender:	SDL
Purpose:	Close the popup currently displayed.

SDL prompts HMI to close the application-related dialog (popup/layout) currently displayed on UI.

7.20.2 Request

7.20.2.1 Behavior

HMI must:

- 1) Close the application-related dialog (popup/screen) initiated by the `methodName` API and currently displayed on UI.
- 2) Respond the request.

Note:

HMI must first reply the `ClosePopUp` method, and then reply the method having been interrupted.

Note:

- The applicable to this RPC result codes are provided in section 7.20.3 Response.
- The sequence diagrams describing the expected HMI behavior are provided in the section 7.20.4 Sequence Diagrams

7.20.2.2 Parameters

Param Name	Type	Mandatory	Description
methodName	String	false	Method that to be closed

7.20.3 Response

Note:

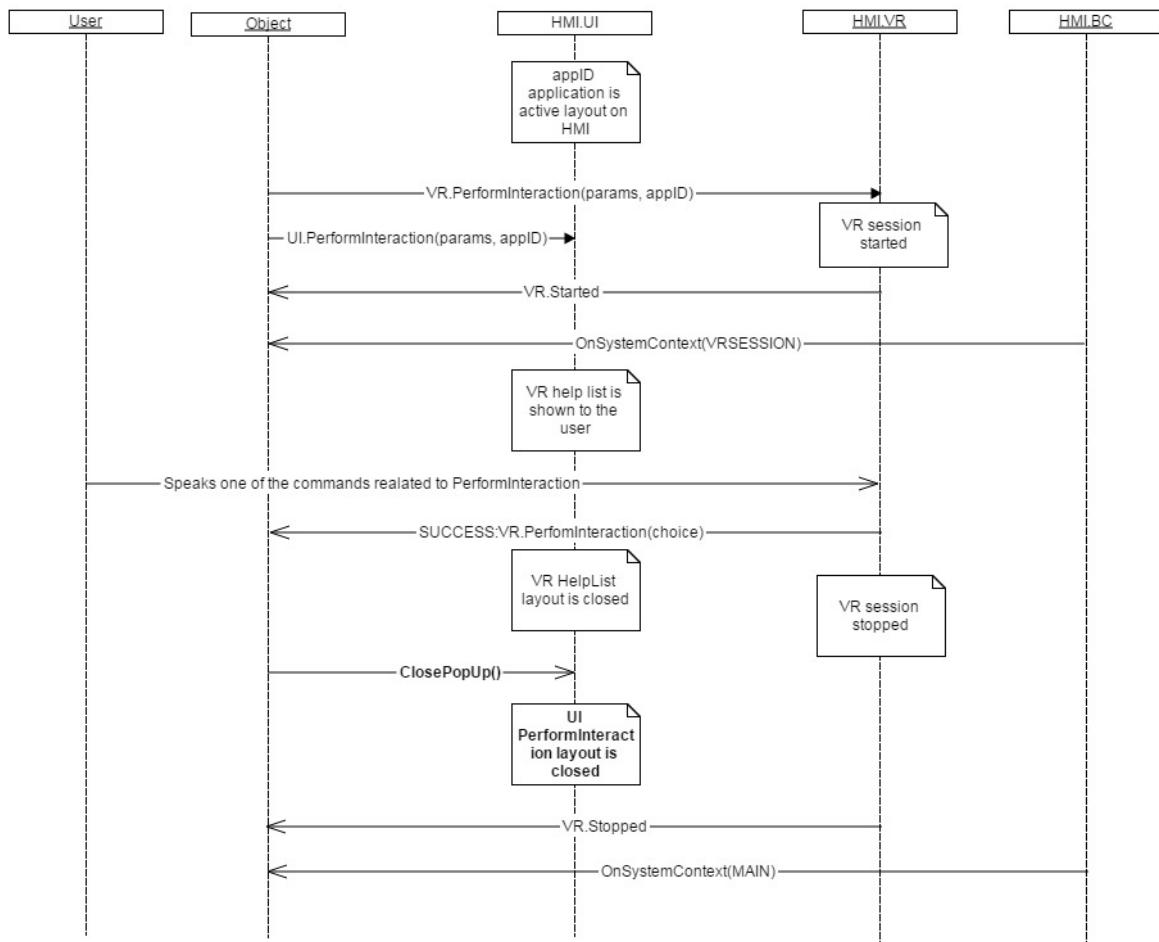
There is a difference in message type for WebSocket and D-Bus connection (see the table below).

Result	Description	Message type		Message Params	Notes
		WebSocket	D-Bus		
Success	SUCCESS: HMI has closed the dialog (popup/screen) displayed on UI at the time the request arrived.	JSON response	Method return	code : 0	
	REJECTED The pop-up on the screen isn't allowed closed by SDL.			Code : 4	
Failure	INVALID_DATA: The data sent is invalid (invalid JSON syntax or parameters out of bounds or of wrong type), there's no pop-up of the named request on the screen now	JSON error message	Method return	code : 11	Applicable for this RPC result codes. Please see Result Enumeration for all SDL-supported codes.
	GENERIC_ERROR: The unknown issue occurred or				

	other codes are not applicable.			
--	---------------------------------------	--	--	--

7.20.4 Sequence Diagrams

7.20.4.1 ClosePopUp for UI.PerformInteraction



7.20.5 JSON Messages Examples

7.20.5.1 Request

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

7.20.5.2 Response

```
{
  "id" : 79,
  "jsonrpc" : "2.0",
```

```

    "result" :
    {
        "code" : 0,
        "method" : "UI.ClosePopUp"
    }
}

```

7.20.5.3 Error message

```

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

```

7.21 OnCommand

7.21.1 Description

Type:	Notification
Sender:	HMI
Purpose:	Inform about the command chosen by the User from UI.

Initially SDL adds the commands to the in-application menu (see `UI.AddCommand`) or sub-menu (see `UI.AddSubMenu`) on UI. The `UI.OnCommand` notification is required by SDL to know the SDL-defined command has been chosen by the User via UI menu.

Note: SDL ignores all invalid notifications which come from HMI (invalid JSON, invalid data types/bounds etc)

HMI must:

Send `UI.OnCommand` notification upon User's choosing an SDL-defined command on UI providing the values of:

- Command ID (`cmdID`) initially provided by SDL via `UI.AddCommand`
- Application ID (`appID`) initially provided by SDL via `OnAppRegistered` or `UpdateAppList`.

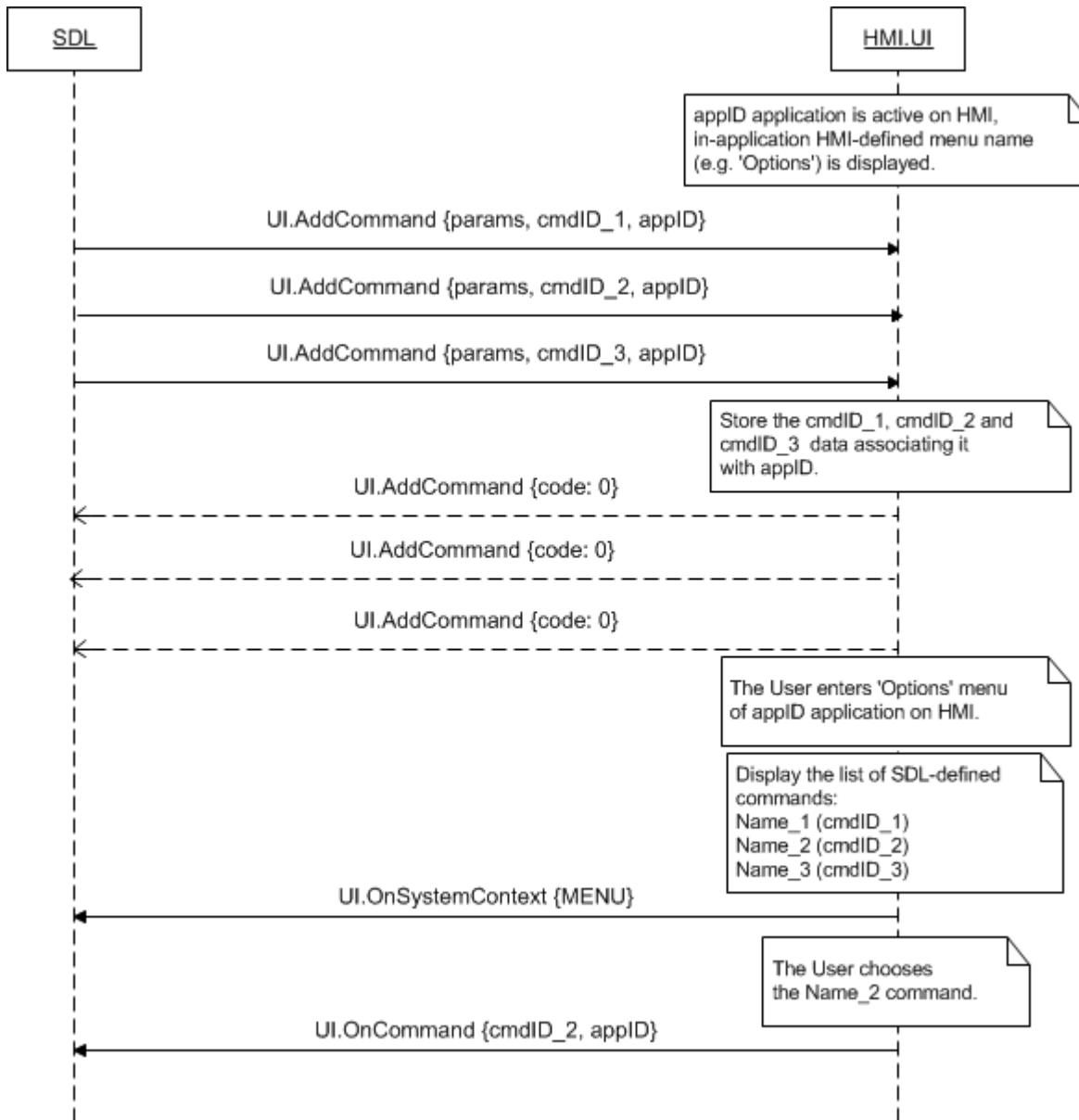
7.21.1.1 Parameters

Param Name	Type	Mandatory	Additional	Description
<code>cmdID</code>	Integer	true	<code>minvalue = 0</code> <code>maxvalue = 2000000000</code>	ID of the command that corresponds to the chosen menu item. This ID was previously sent by SDL within <code>AddCommand</code> request.

appID	Integer	true	-	ID of the application that concerns this RPC. Initially is provided by SDL via OnAppRegistered or UpdateAppList.
-------	---------	------	---	--

7.21.2 Sequence Diagrams

7.21.2.1 OnCommand



7.21.3 JSON Messages Examples

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

```
    }  
}
```

7.22 OnSystemContext

7.22.1 Description

Type:	Notification
Sender:	HMI
Purpose:	Inform about the User-initiated interaction on HMI.

SDL requires the information about the application state on HMI and whether the User-initiated interaction is currently taking place on it.

Note: SDL ignores all invalid notifications which come from HMI (invalid JSON, invalid data types/bounds etc)

HMI must:

Send the OnSystemContext notification of:

- MAIN: when the User doesn't interact with application via persistant display (neither the in-application menu nor dialog/overlay is currently displayed).
- VRSESSION: when VR becomes active
 - as a result of PTT button press and help list of VR synonyms being displayed
 - as a result of UI.PerformInteraction (of BOTH or VR_ONLY mode) request from SDL (when VR session is active)
- MENU: when the User enters in-application menu (e.g. 'Options') on UI.
- HMI_OBSCURED: when HMI is currently obscuring the application persistent display with:
 - The dialog of ScrollableMessage, Slider, PerformInteraction (MANUAL_ONLY and BOTH mode), PerformAudioPassThru, etc
 - Any HMI-defined (system) dialog.
- ALERT: when SDL-requested Alert dialog is displayed

OnSystemContext related to successful Alert: 'appId'

parameter optionality

1. HMI ought to not send AppID for MENU and HMI_OBSCURED system contexts. Only the apps in FULL may get these updates.
- 2 .In case HMI sends OnSystemContext of MENU and HMI_OBSCUREDwith appId by HMI, appId is ignored and anyway the notification is transferred to the app in FULL HMI Level
3. HMI must send OnSystemContext with appId parameter for MAIN and ALERT values. In case there's no such application with appId received, the notifications will be ignored by SDL

Note:

The OnSystemContext must be sent right after the event has occurred.

Each moment of time, SystemContext value of an application must take one of Common.SystemContext values.

7.22.1.1 Parameters

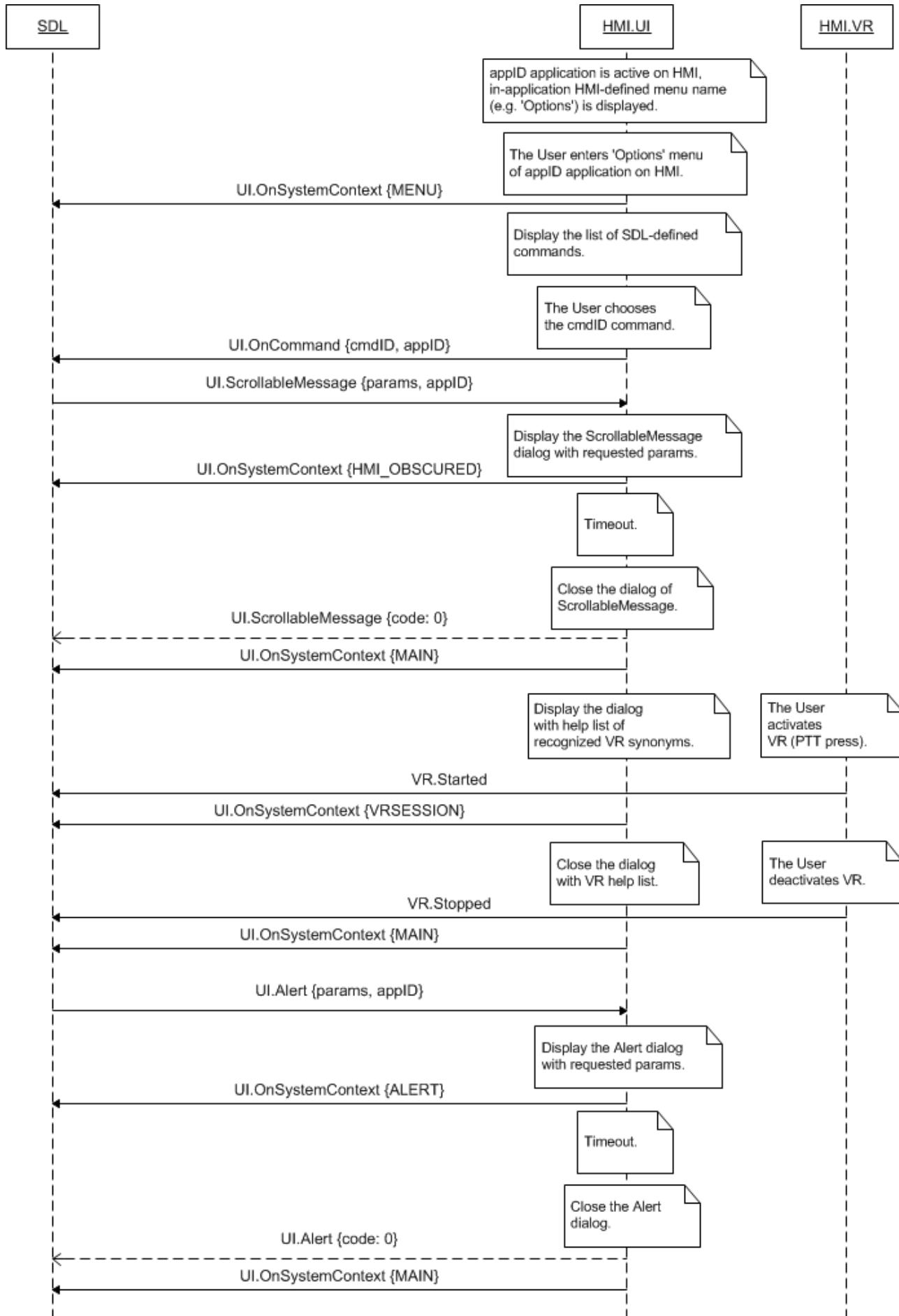
Param Name	Type	Mandatory	Description
systemContext	Common.SystemContext	true	The context the application is brought into. See SystemContext.
appID	integer	false	

7.22.1.2 SystemContext

Element name	Value	Short Description
MAIN	0	If there is currently no user interaction (user-initiated or app-initiated) with the HMI, HMI must notify SDL that SystemContext is MAIN.
VRSESSION	1	Must be sent if there is a current VR-oriented user interaction (VR becomes active as a result of PTT or PerformInteraction).
MENU	2	Must be sent if HMI is currently displaying an in-application menu onscreen.
HMI_OBSCURED	3	Must be sent if HMI is currently obscuring the application display either with a system or with application overlay (except of Alert element).
ALERT	4	Must be sent if the Alert message is currently displayed onscreen.

7.22.2 Sequence Diagrams

7.22.2.1 *OnSystemContext for different HMI states*



7.22.3 JSON Messages Examples

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

7.23 OnLanguageChange

7.23.1 Description

Type:	Notification
Sender:	HMI
Purpose:	Inform about the UI language is changed.

SDL needs to be in the know when the User changes the HMI display language: Upon the receipt of OnLanguageChange notification SDL will unregister all applications of different language to provide them with possibility to re-register with the correct (new HMI) display language.

Note: SDL ignores all invalid notifications which come from HMI (invalid JSON, invalid data types/bounds etc)

HMI must:

- 1) Send the UI.OnLanguageChange notification when the User switches HMI to another language and provide this new value with the `language` parameter.

7.23.1.1 Parameters

Param Name	Type	Mandatory	Description
language	Common.Language	true	Language that UI has switched to.

7.23.1.2 Language

Element 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

Element Name	Value	Description
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
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 Belgium (Flemish)
EL-GR	25	Greek
HU-HU	26	Hungarian
FI-FI	27	Finnish
SK-SK	28	Slovak

7.23.2 Sequence Diagrams

7.23.2.1 OnLanguageChange

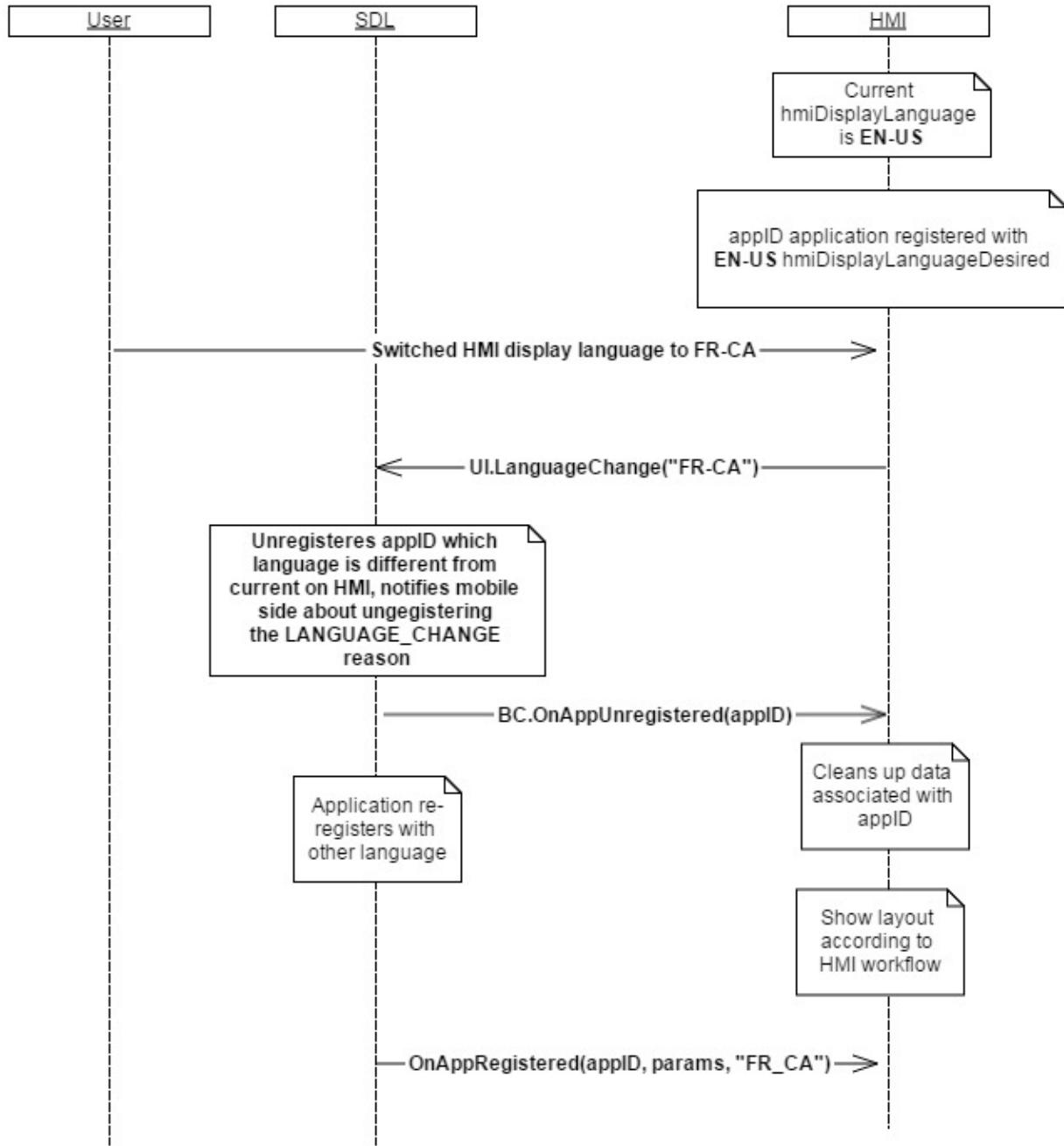
Note:

In case of several applications registered with HMI, upon *OnLanguageChange* receipt *SDL* will send the following to HMI:

- *OnAppUnregistered notifications for each of appIDs with language different to new one on HMI display.*

After the applications re-register with the new (provided via *OnLanguageChange*)

hmiDisplayLanguageDesired, *SDL* will send the corresponding *OnAppRegistered* notifications to HMI.



7.23.3 JSON Messages Examples

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

7.24 OnKeyboardInput

7.24.1 Description

Type:	Notification
Sender:	HMI
Purpose:	Inform about the keyboard event.

HMI may be requested by SDL to display the touchscreen keyboard via:

- PerformInteraction with layoutMode: KEYBOARD
 - keyboard to be displayed
- PerformInteraction with layoutMode:
ICONS_WITH_SEARCH – choices as icons to be displayed, the keyboard to be displayed upon User's clicking in search bar area.
- PerformInteraction with layoutMode:
LIST_WITH_SEARCH – choices as a list to be displayed, the keyboard to be displayed upon User's clicking in search bar area.

HMI is expected to display a keyboard in case initiated by the user:

- as a result of appropriate button press at the 'navigation' template
- as the result of recognized VR command that requests a keyboard search at the 'navigation' template

HMI expected behavior:

OnKeyboardInput(ENTRY_VOICE), reflects that the "voice" button was pressed on the screen and the user uses voice search instead of typing. Keyboard must be closed on this event. For more details see [7.24.2.5](#)

OnKeyboardInput for ENTRY_VOICE diagram

Note: ENTRY_VOICE is **not** related to UI.PerformInteraction(KEYBOARD).

The keyboard layout, language, way of sending the entry are defined by SDL via UI.SetGlobalProperties (keyboardProperties structure) RPC.

Depending on keypressMode value (from keyboardProperties structure of UI.SetGlobalProperties), HMI must send the onKeyboardInput notification with the following data:

- SINGLE_KEYPRESS: each and every User's keypress must be reported (new notification for every newly entered single symbol).
- QUEUE_KEYPRESSES: the whole entry is reported only after the User submits it (by 'Search' button click displayed on touchscreen keyboard)
- RESEND_CURRENT_ENTRY: the whole entry must be reported each and every time the User makes a new keypress (new notification with all previously entered symbols and a newly entered one appended).

HMI must:

- 1) Send the OnKeyboardInput notification providing the data entered from and the event occurred over the keyboard.

Note: SDL ignores all invalid notifications which come from HMI (invalid JSON, invalid data types/bounds etc)

7.24.1.1 Parameters

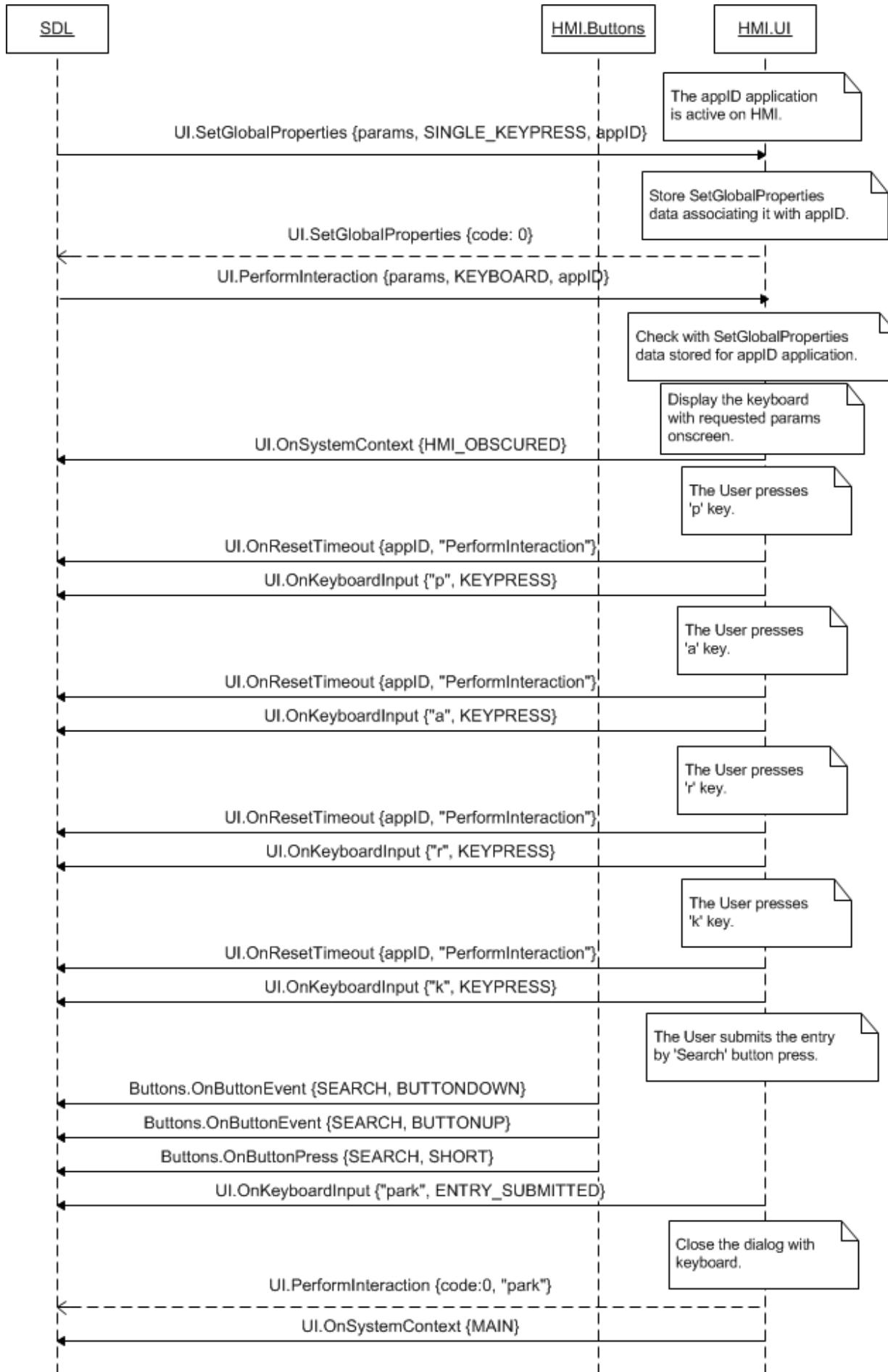
Param Name	Type	Mandatory	Additional	Description
event	Common.KeyboardEvent	true	-	The event occurred over the touchscreen keyboard.
data	String	false	Maxlength = 500 Minlength = 0	On-screen keyboard input data: - A single symbol in case keypressMode is defined as SINGLE_KEYPRESS via UI.SetGlobalProperties. - The whole entry after the User has pressed 'Search' button in case keypressMode is defined as QUEUE_KEYPRESS via UI.SetGlobalProperties. - The whole entry every time the User makes a new keypress in case keypressMode is defined as RESEND_CURRENT_ENTRY via UI.SetGlobalProperties.

7.24.1.2 KeyboardEvent

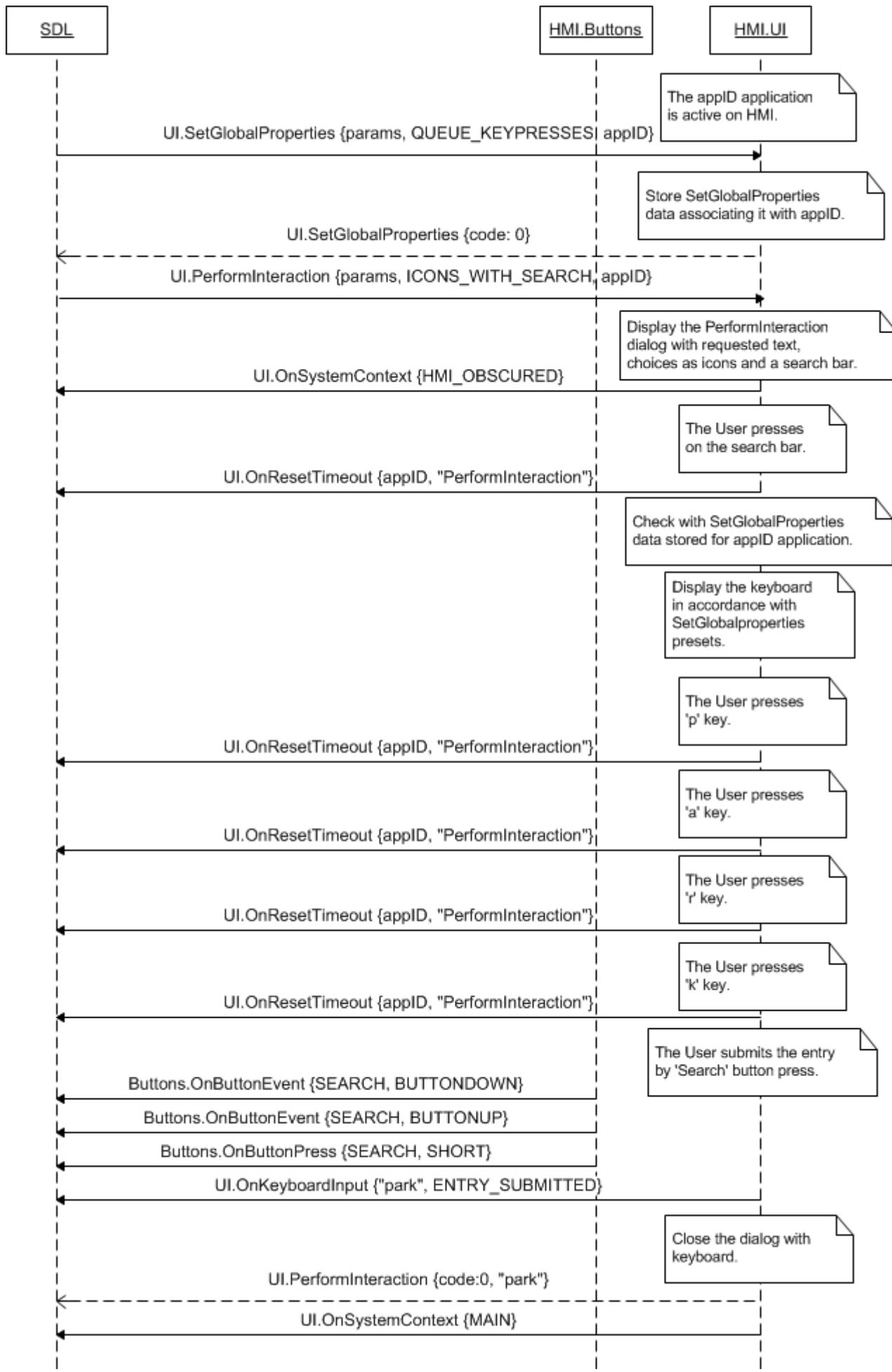
Element name	Short Description
KEYPRESS	The User has pressed the keyboard key (applies to both SINGLE_KEYPRESS and RESEND_CURRENT_ENTRY modes).
ENTRY_SUBMITTED	The User has finished entering text from the keyboard and submitted the entry.
ENTRY_VOICE	Indicates that a voice button was pressed for a request to search via voice
ENTRY_CANCELLED	The User has pressed the HMI-defined 'Cancel' button.
ENTRY_ABORTED	The User has not finished entering text and the keyboard is aborted with the event of higher priority.

7.24.2 Sequence Diagrams

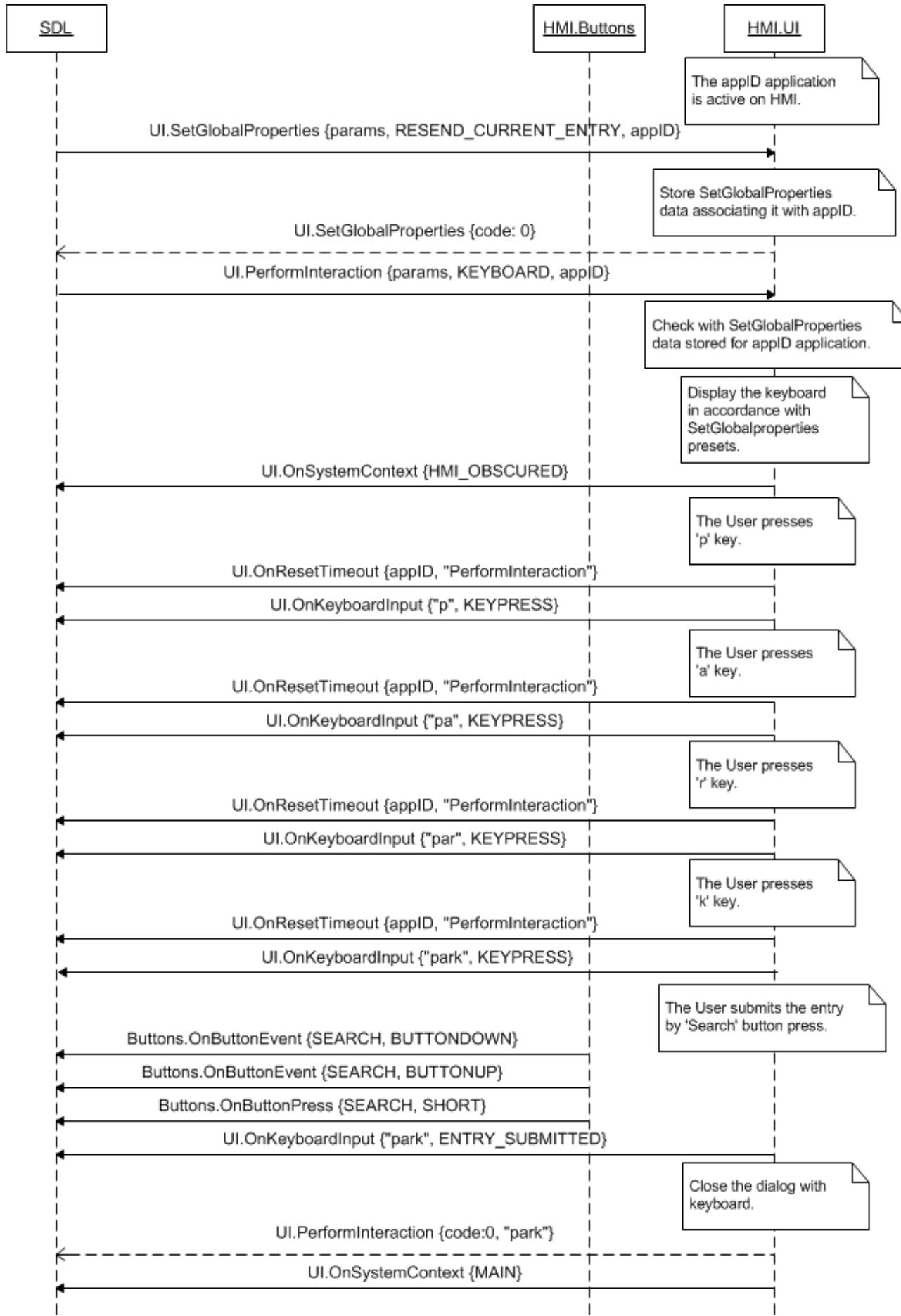
7.24.2.1 OnKeyboardInput for SINGLE_KEYPRESS keypressMode



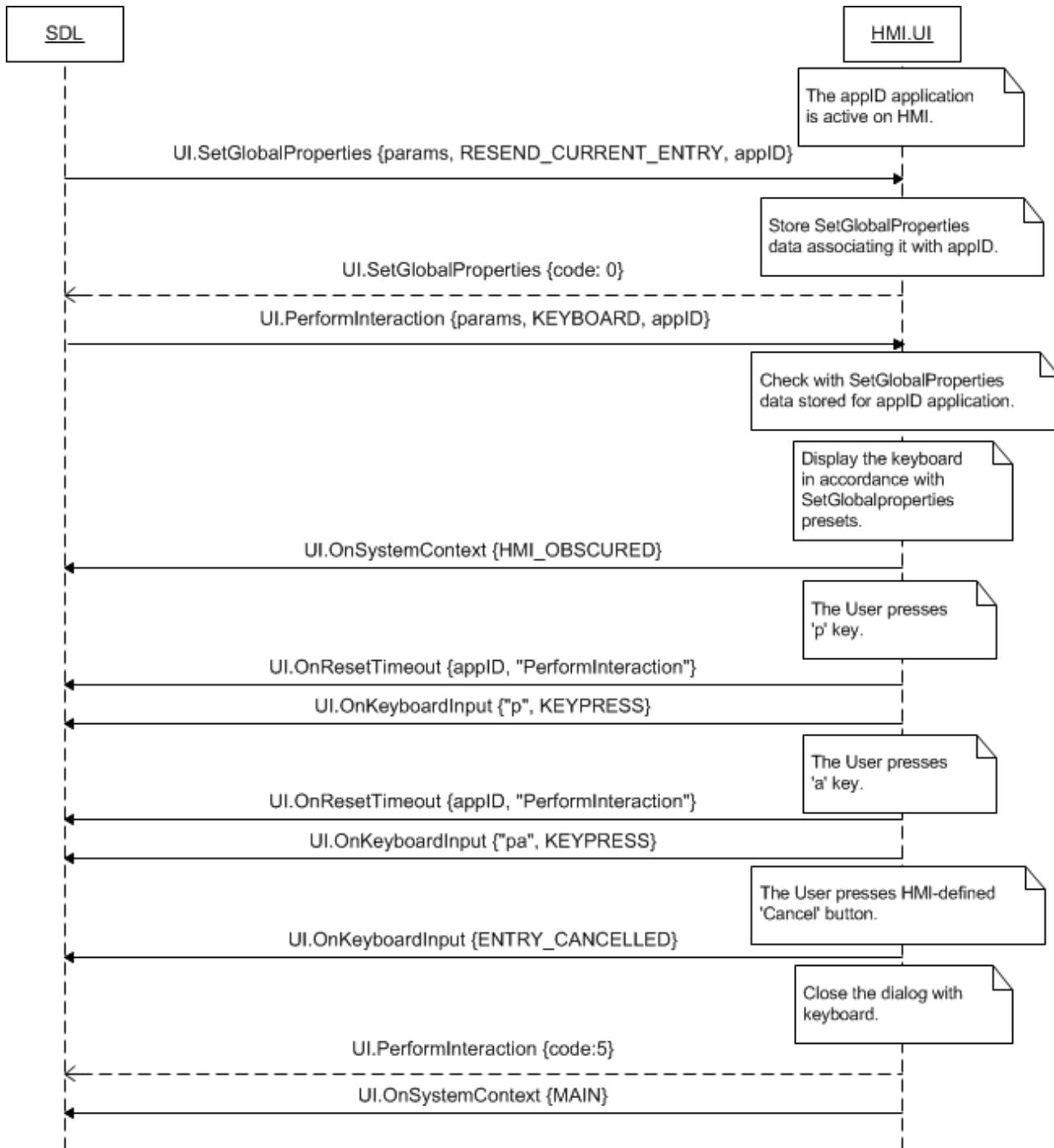
**7.24.2.2 *OnKeyboardInput* for
QUEUE_KEYPRESSES keypressMode**



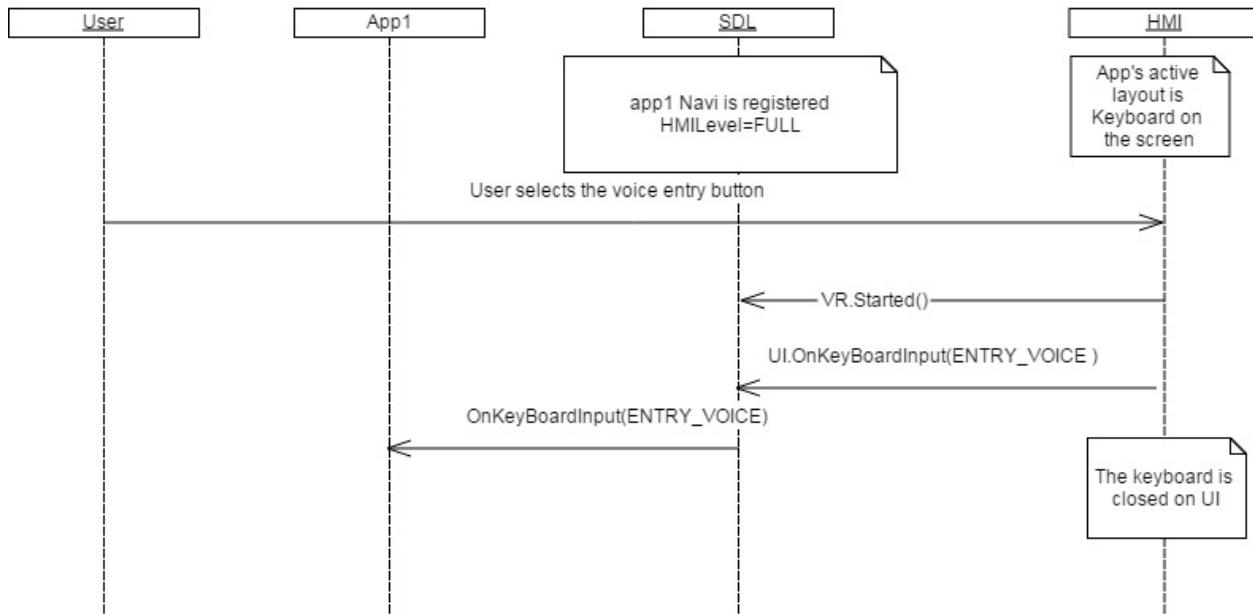
**7.24.2.3 *OnKeyboardInput* for
RESEND_CURRENT_ENTRY keypressMode**



7.24.2.4 OnKeyboardInput for cancelled entry



7.24.2.5 OnKeyboardInput for ENTRY_VOICE



7.24.3 JSON Messages Examples

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

7.25 OnTouchEvent

7.25.1 Description

Type:	Notification
Sender:	HMI
Purpose:	Inform the touch event has occurred.

SDL needs to be informed about every User's touching of the touch screen during the navi video streaming is active screen layout.

- SDL supports up to 10 touches in a one touch event (this covers touch and multitouch events at the same time),
- When a single touch happens, "event" parameter contains the only element in the array, in case other fingers joined, an array contains the event data for every finger.

HMI must:

- 1) Notify the user about touch events on the navi streaming screen via OnTouchEvent notifications during all active navi streaming for the application in focus.

- 2) For MOVE TouchType OnTouchEvent is sent for every coordinate got while the user is moving a finger on the screen (means a single element in "c" parameter array).
- 3) In case any type touch event has been performed for more than one finger, the notification must contain TouchEvent data for each finger took part in the touch.

Note: SDL ignores all invalid notifications which come from HMI (invalid JSON, invalid data types/bounds etc)

7.25.1.1 Parameters

Param Name	Type	Mandatory	Additional	Description
type	Common.TouchType	true	-	The type of touch event.
event	Common.TouchEvent	true	array = true minsize = 1 maxsize = 10	List of all individual touches involved in this event.

7.25.1.2 TouchEvent

Param Name	Type	Mandatory	Additional	Description
id	Integer	true	minvalue = 0 maxvalue = 9	A touch's unique identifier. The application can track the current touch events by id. If a touch event has type begin, the id should be added to the set of touches. If a touch event has type end, the id should be removed from the set of touches.
ts	Integer	true	Array = true minvalue = 0 maxvalue = 2147483647 minsize = 1 maxsize = 1000	The time that the touch was recorded. This number can be the time since the beginning of the session or something else as long as the units are in milliseconds. The timestamp is used to determine the rate of change of position of a touch. The application also uses the time to verify whether two touches, with different ids, are part of a single action by the user.
c	Common.TouchCoord	true	Array = true minsize = 1 maxsize = 1000	The coordinates of the screen area where the touch event occurred.

7.25.1.2 TouchCoord

Param Name	Type	Mandatory	Additional	Description
x	Integer	true	minvalue = 0 maxvalue = 10000	The x coordinate of the touch.

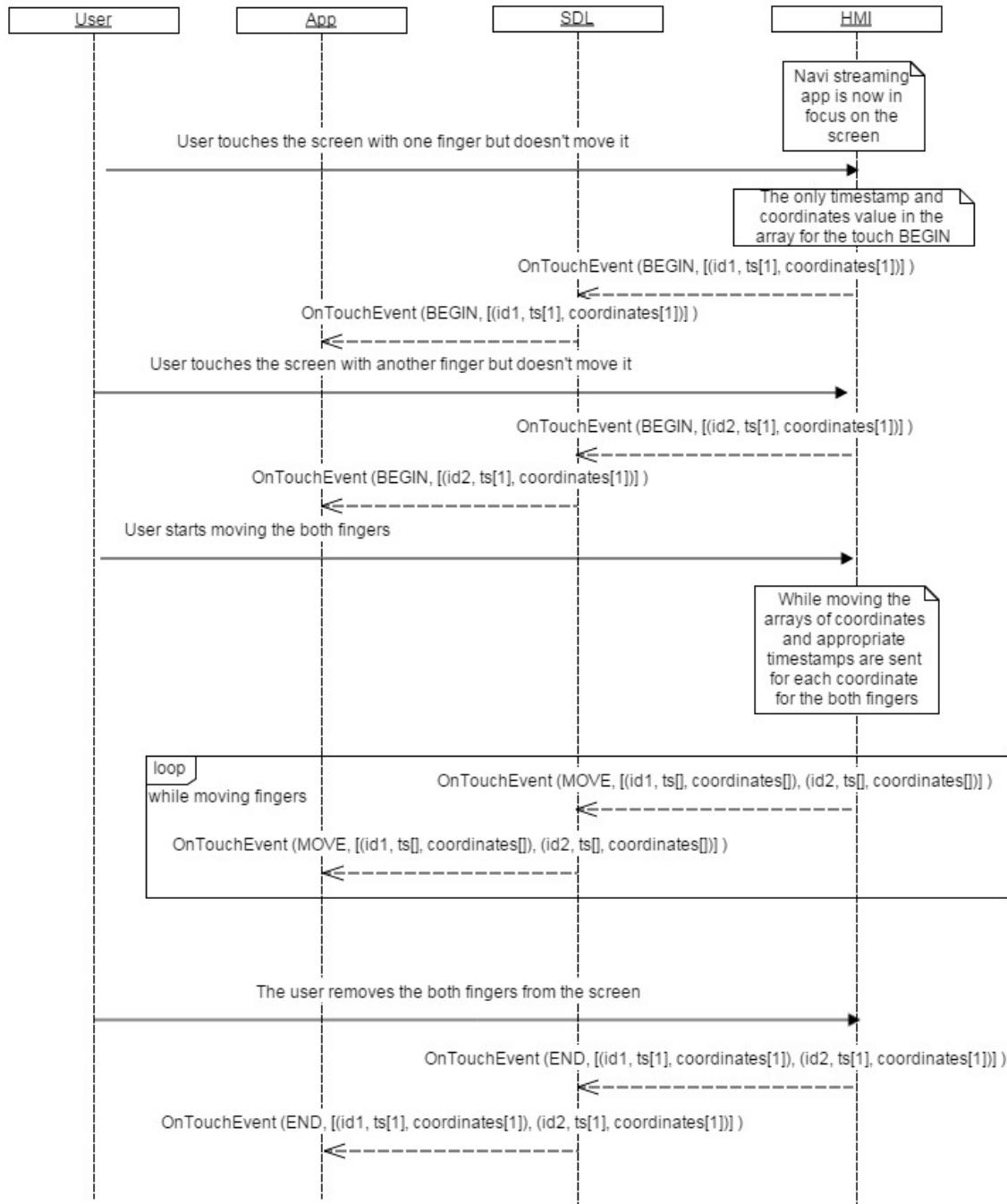
Y	Integer	true	minvalue = 0 maxvalue = 10000	The y coordinate of the touch.
---	---------	------	----------------------------------	--------------------------------

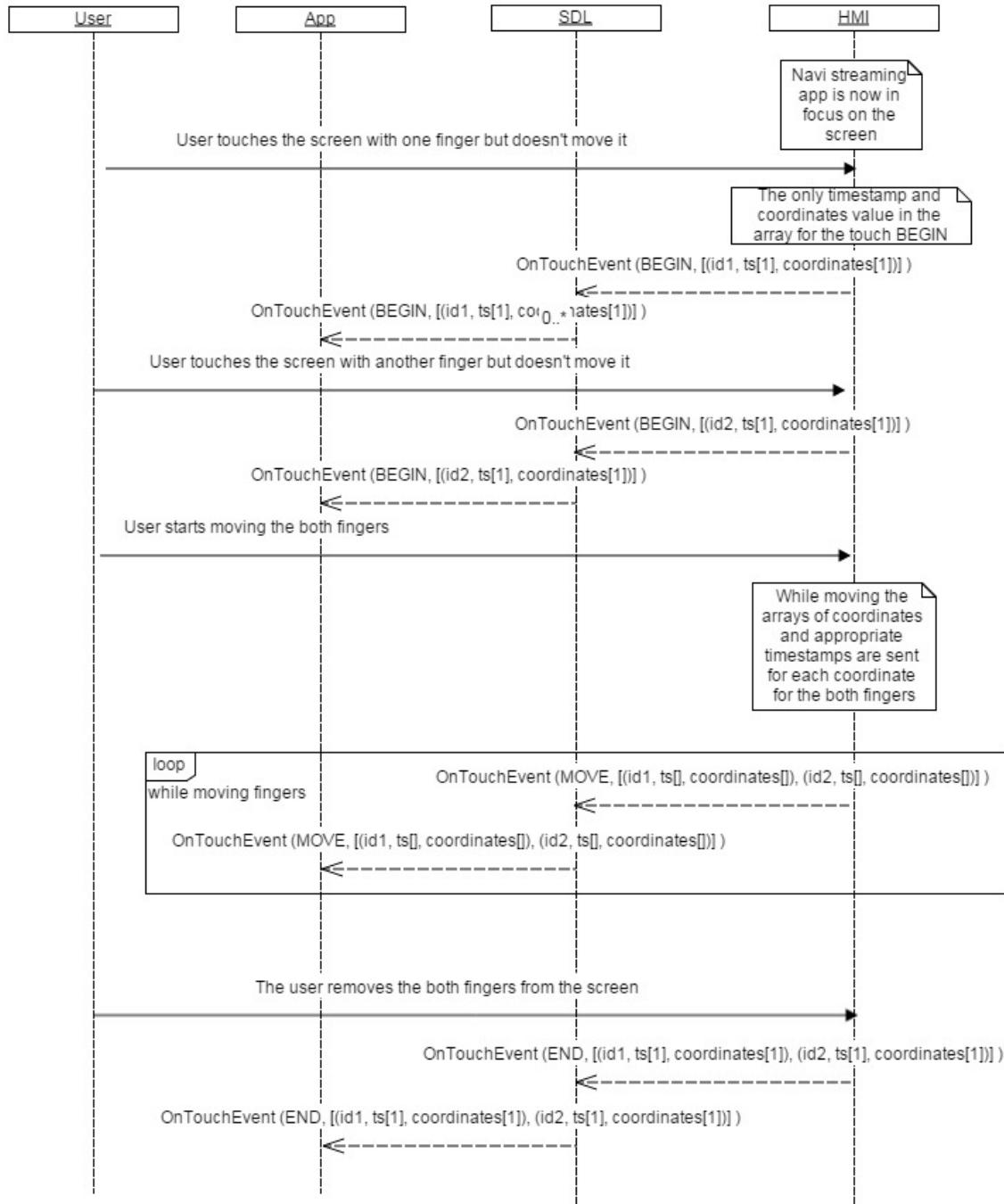
7.25.1.2 TouchType

Element name	Value	Short Description
BEGIN	0	The user has touched the screen.
MOVE	1	The User has moved his finger over the screen.
END	2	The User has removed his finger from the screen.

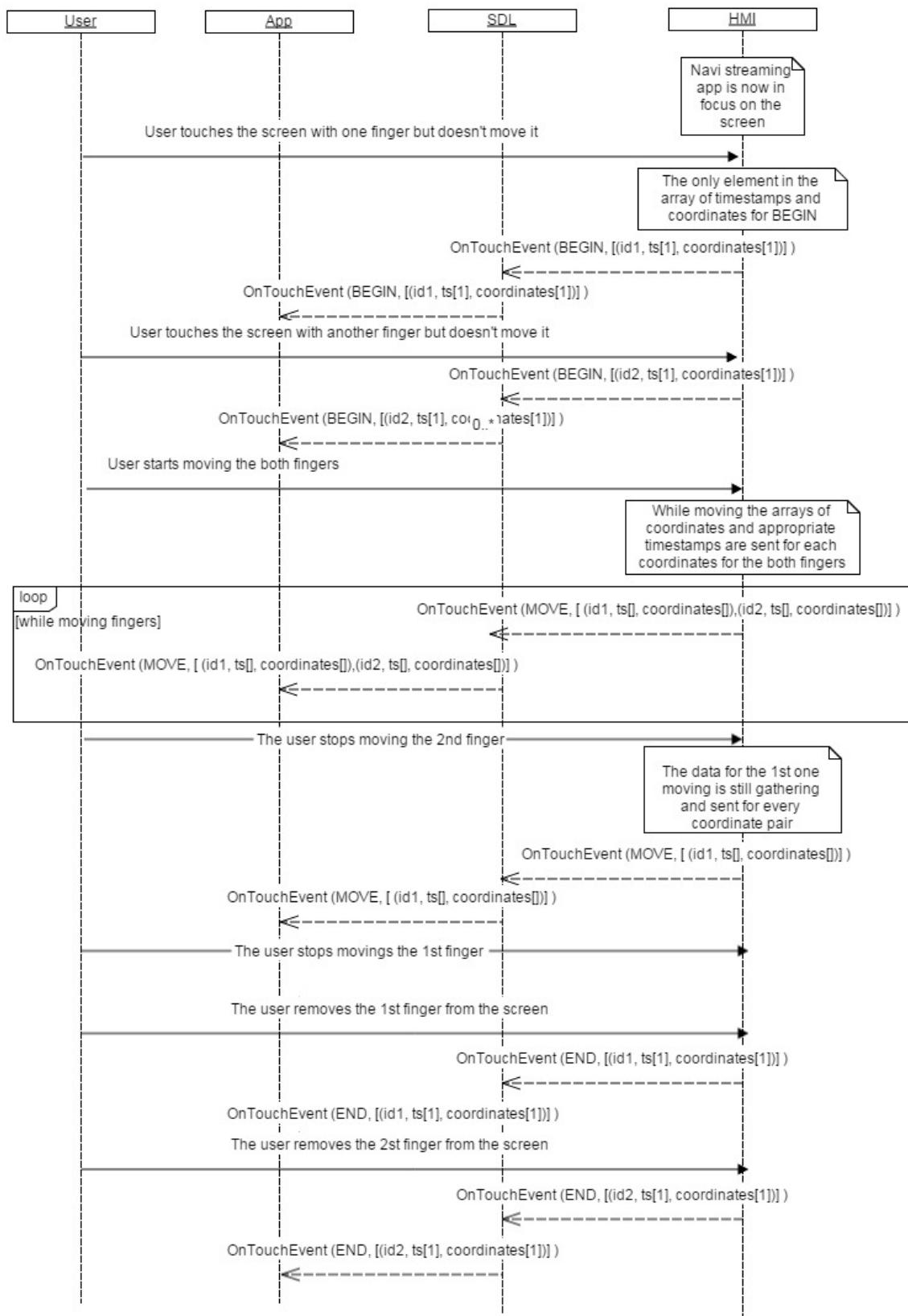
7.25.2 Sequence Diagrams

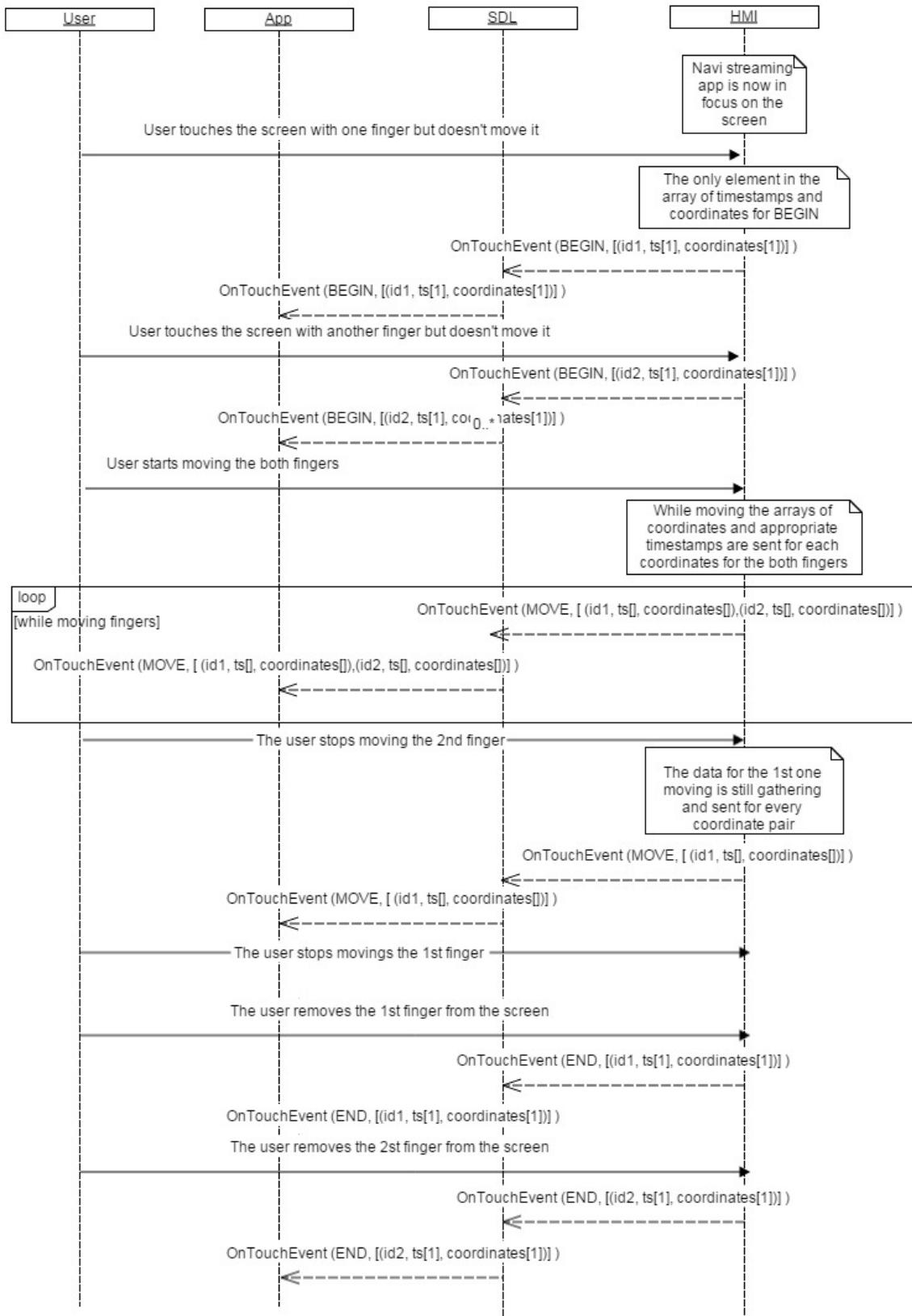
7.25.2.1 OnTouchEvent moving two fingers stopped together





**7.25.2.2 OnTouchEvent moving two fingers,
one stopped earlier**





7.25.3 JSON Messages Examples

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

7.26 OnResetTimeout

7.26.1 Description

Type:	Notification
Sender:	HMI
Purpose:	Inform about timeout reset for the named function.

SDL needs to be in the know when the timeout of the recently called UI RPC is reset by definite User's actions. This is required to prevent sending a response to mobile side on currently performing request. Usually SDL sends a response by timeout to mobile application if it doesn't get a response from HMI within a request defined timeout or SDL defined timeout,

HMI must:

Send the OnResetTimeout notification in the following cases :

- 1) PerformInteraction of KEYBOARD layoutMode – upon every User's keypress.
- 2) ScrollableMessage:
 - Upon User's actions over the message (e.g. scrolling)
 - Upon KEEP_CONTEXT soft button (defined within ScrollableMessage RPC) press.
- 3) Alert – upon KEEP_CONTEXT soft button (defined within Alert RPC) press

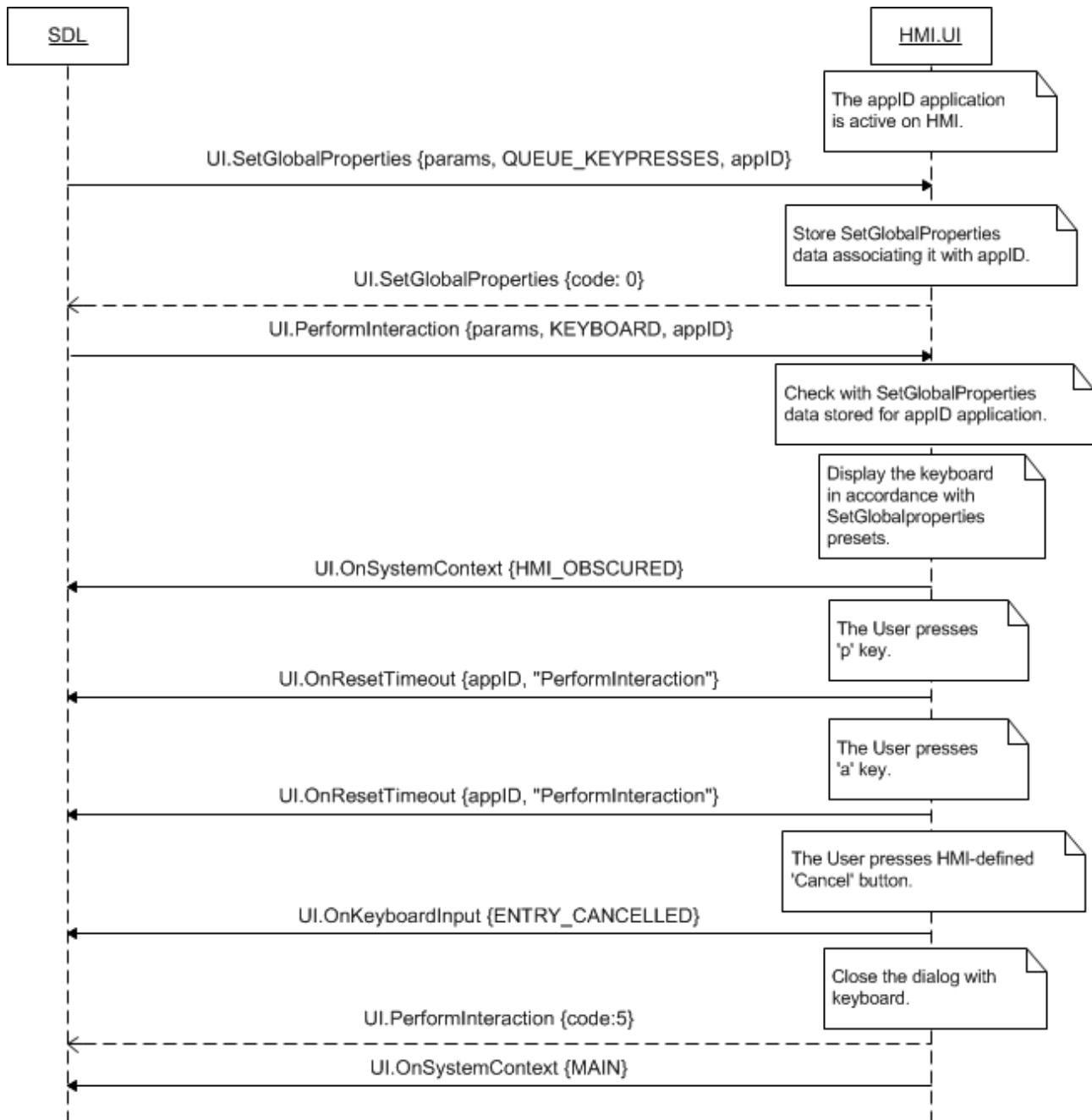
Note: SDL ignores all invalid notifications which come from HMI (invalid JSON, invalid data types/bounds etc)

7.26.1.1 Parameters

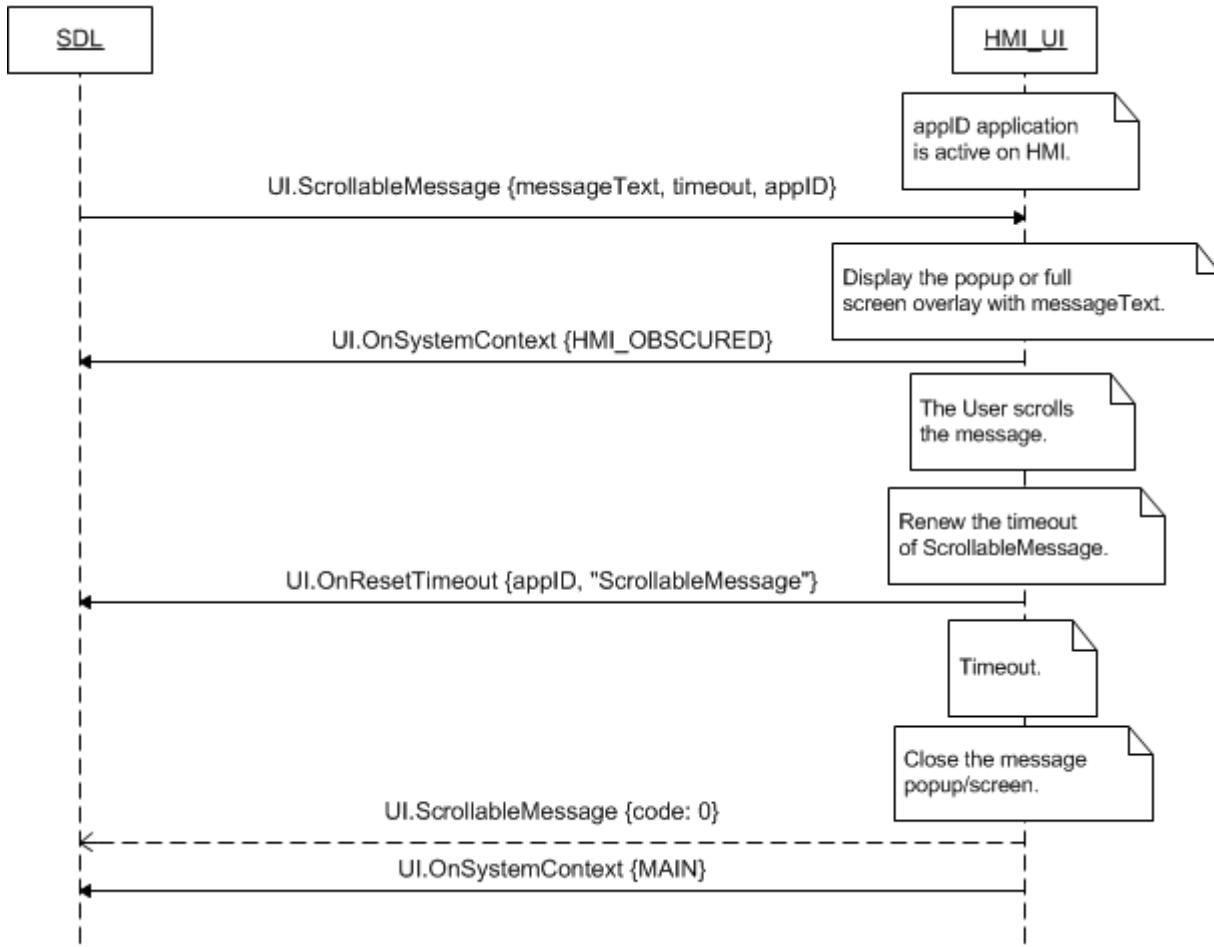
Param Name	Type	Mandatory	Description
appID	Integer	true	Id of application that is related to notification.
methodName	String	true	Currently used method name on which was triggered action

7.26.2 Sequence Diagrams

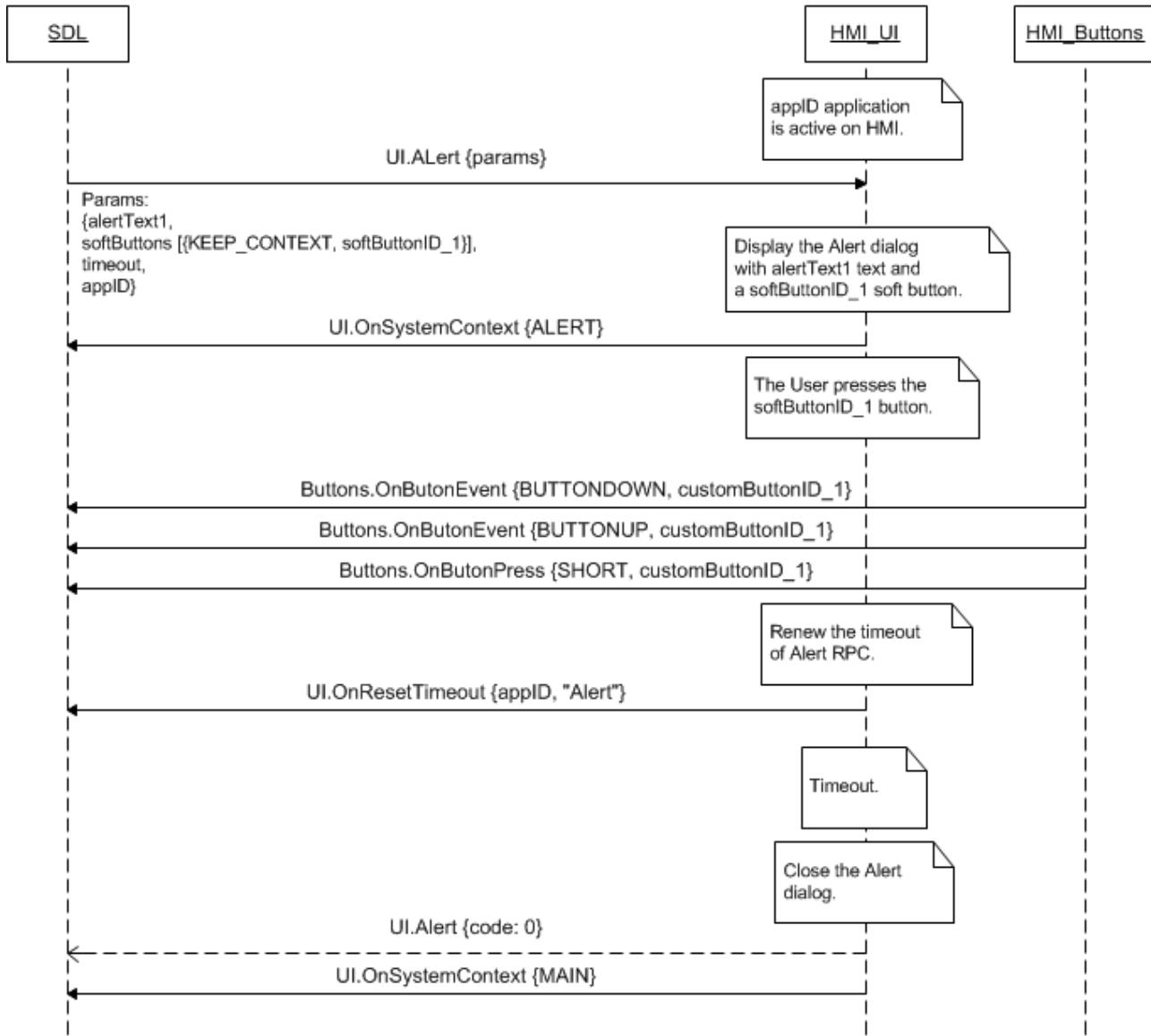
7.26.2.1 OnResetTimeout upon keypress during PerformInteraction (KEYBOARD)



7.26.2.2 OnResetTimeout upon User's scrolling the message during ScrollableMessage



7.26.2.2 OnResetTimeout upon KEEP_CONTEXT soft button press during Alert



7.27.3 JSON Messages Examples

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

7.27 OnDriverDistraction

7.27.1 Description

Type:	Notification
Sender:	HMI
Purpose:	Inform about changes of driver distraction state.

HMI must:

- 1) Inform SDL via OnDriverDistraction whenever the driver distraction mode is activated/deactivated on HMI.

Driver distraction rules may be specific to country/area, so it depends on HMI when to trigger activate/deactivate states.

Note: SDL ignores all invalid notifications which come from HMI (invalid JSON, invalid data types/bounds etc)

7.27.1.1 Parameters

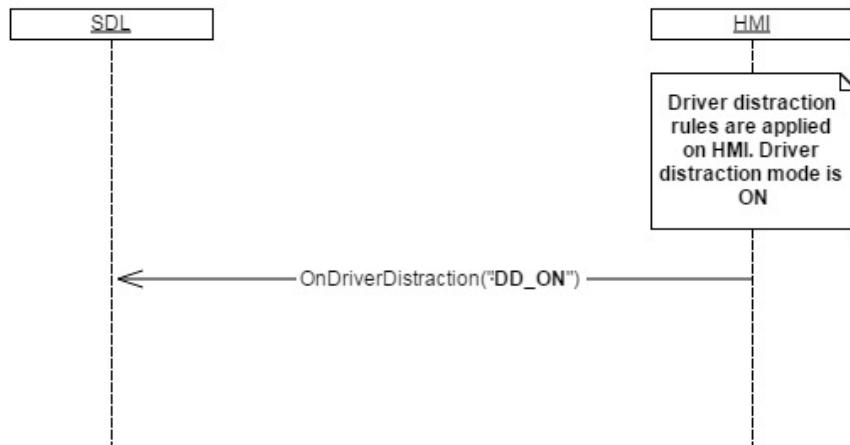
Param Name	Type	Mandatory	Description
state	Common.DriverDistractionState	true	Notifies the application of the current driver distraction state

7.27.1.2 DriverDistractionState

Element name	Value	Short Description
DD_ON	0	Driver Distraction rules are in effect.
DD_OFF	1	Driver Distraction rules are not in effect.

7.27.2 Sequence Diagrams

7.27.2.1 OnDriverDistraction notification



7.27.3 JSON Messages Examples

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

7.28 OnRecordStart

7.29.1 Description

Type:	Notification
Sender:	SDL
Purpose:	Start capturing audio data from the microphone

There're 2 conditions when SDL sends the notification for starting recording while PerformAudioPassThru is performing on HMI:

- Right after TTS.Speak portion of PerformAudioPassThru ended on HMI ([see 7.28.2.1 diagram](#))
- In case just UI.PerformAudioPassThru portion should be performed, the notification must be send by SDL right after UI.PerformAudioPassThru has been sent to HMI.

HMI must:

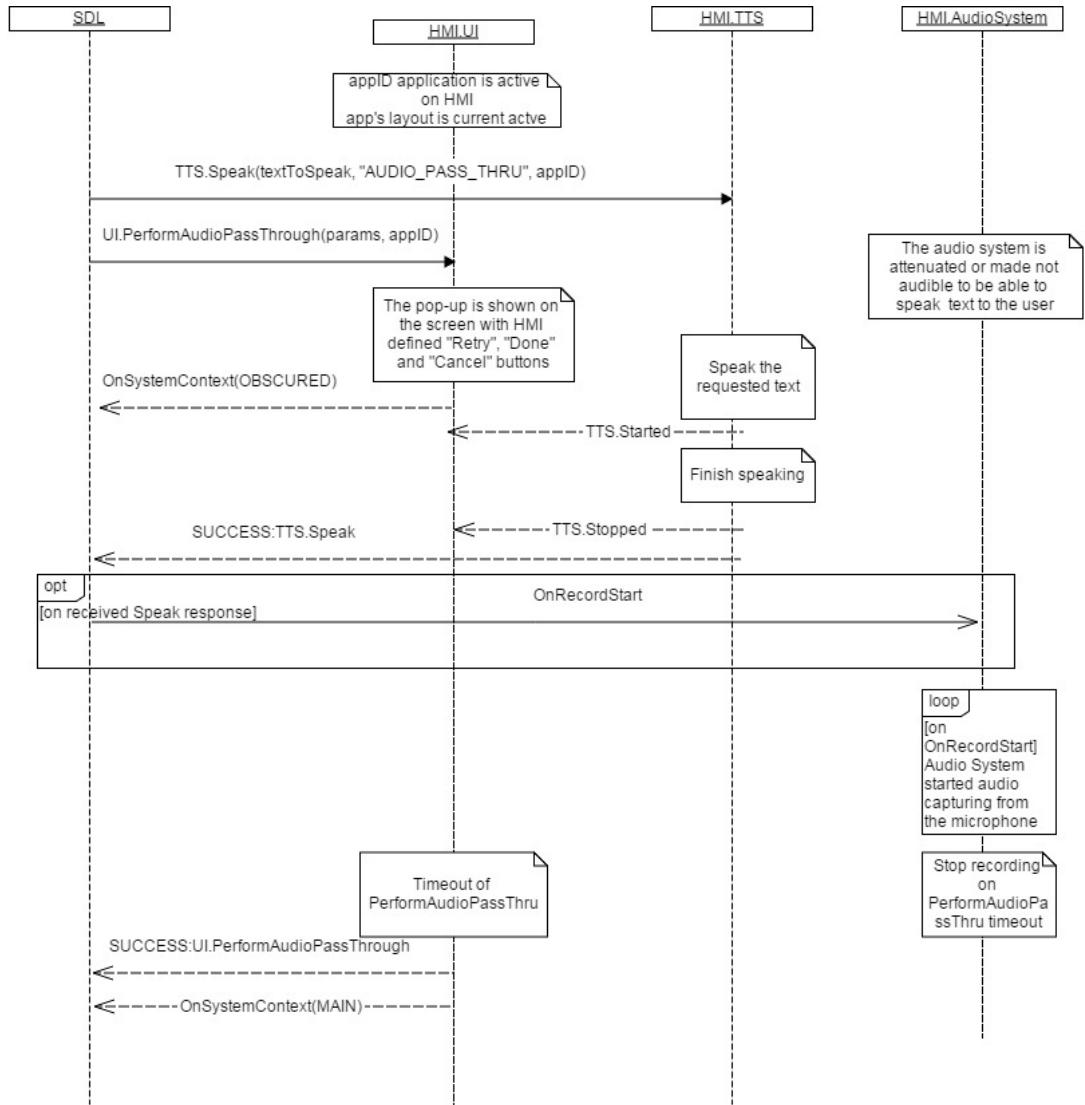
- 1) Start capturing data from the microphone as soon as get OnRecordStart. Audio data must be written into the directory on a platform and available to SDL (one directory up above SDL folder)

7.29.1.1 Parameters

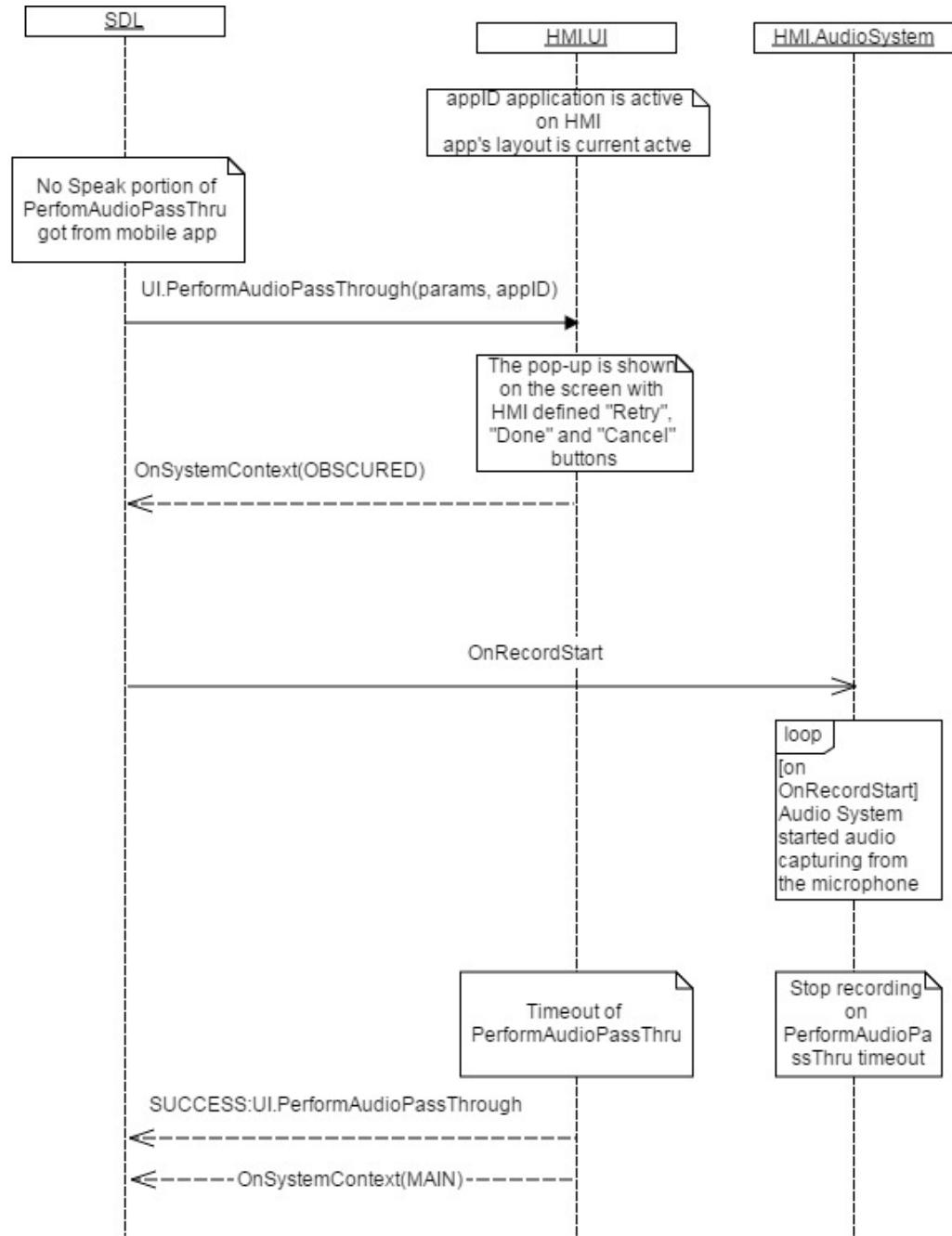
Param Name	Type	Mandatory	Description
appID	Integer	true	ID of application related to this RPC.

7.28.2 Sequence Diagrams

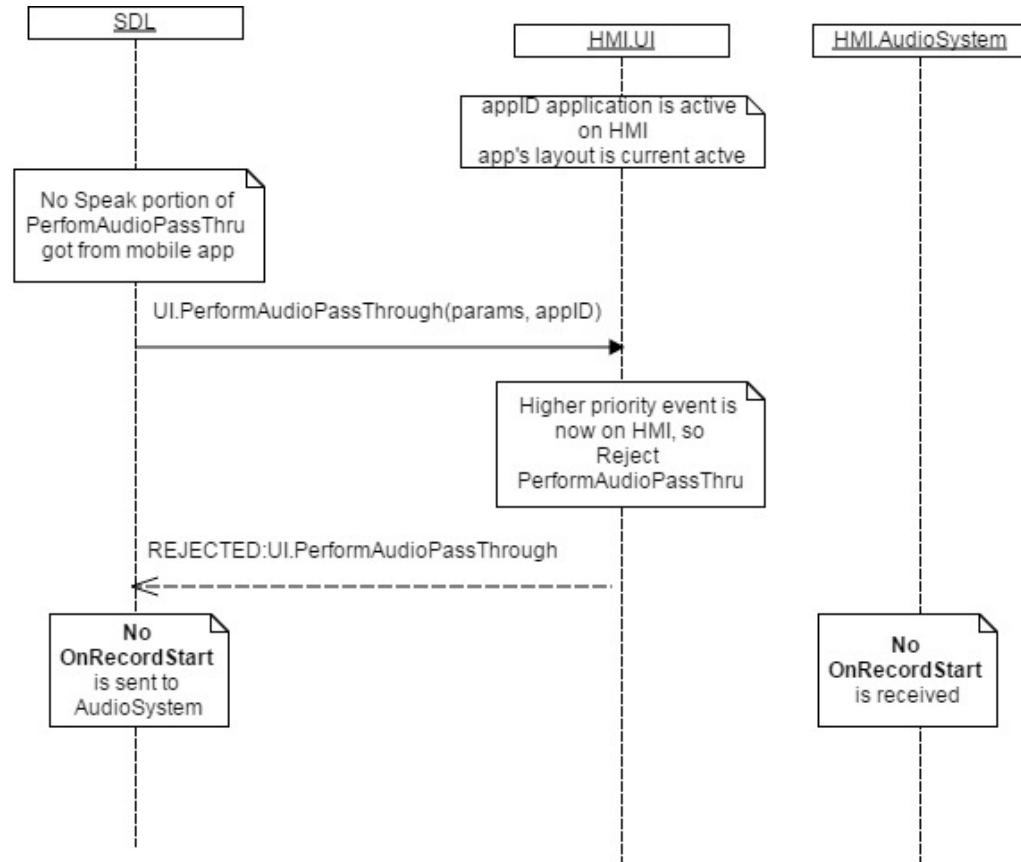
7.28.2.1 OnRecordStart with TTS.Speak



7.28.2.2 OnRecordStart without TTS.Speak



7.28.2.3 OnRecordStart NOT sent if UI.PerformAudioPassThru is REJECTED



7.28.3 JSON Messages Examples

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

7.29 SetDisplayLayout

7.29.1 Description

Type:	Function
Sender:	SDL
Purpose:	Set-up HMI-predefined display template for the named application.

With `SetDisplayLayout` SDL requests HMI to use the named template when arranging the display layout of the named application.

Within SetDisplayLayout request SDL will use the template name (see section 7.29.2.2 Parameters) that HMI previously provides within response to UI.GetCapabilities (see section [7.2 Response](#) and the below extract of corresponding JSON message).

Extract of JSON response to UI.GetCapabilities that contains templates names
<pre>"method" : "UI.GetCapabilities", "params": { ... "displayCapabilities" : { ... "templatesAvailable" : ["template_1", "template_2"], ... }, ... }</pre>

7.29.2 Request

7.29.2.1 Behavior

HMI must:

1. Check whether the displayLayout parameter (see table in section 7.29.2.2) contains the correct template name (string value):
 - If correct, proceed with steps 2,3,4.
 - If not correct or some error occurred, proceed with step 3 only.
2. Associate the named template of displayLayout with the provided application appID (see table in section 7.29.2.2).
3. Respond with one of the appropriate result codes (see section 7.29.3 Response for applicable result codes). And in case of SUCCESS return the capabilities of the named template:
 - displayCapabilities: the type of display, supported text and image fields, supported formats of media clock and other that is described in [section 7.29.3.2](#).
 - buttonCapabilities: capabilities of hardware buttons: the names of available buttons, information about whether notifications on short/long presses and up/down events are supported by HMI (see [section 7.29.3.13](#)).
 - softButtonCapabilities: capabilities of soft buttons: information about whether HMI supports referencing image and notifying on short/long presses and up/down events (see [section 7.29.3.15](#)).
 - presetBankCapabilities: information about whether HMI supports duplicating the hardware custom presets with onscreen buttons (see [section 7.29.3.18](#)).

4. When the appID application is activated on HMI, HMI must switch to the corresponding template (associated in step 2.) that implements the corresponding capabilities (reported in step 3.)

Note:

- The applicable to this RPC result codes are provided in section 7.29.3 Response.
- The sequence diagrams describing the expected HMI behavior are provided in the section 7.29.4 Sequence Diagrams

7.29.2.2 Parameters

Param Name	Type	Mandatory	Additional	Description
displayLayout	String	true	Maxlength = 500	The name of predefined or dynamically created screen layout. Currently only predefined screen layouts are defined. See section 7.29.3.19 PredefinedLayout for the variants of template names. Still, HMI is not limited with these values and is free to create the name(s) that fit the template behavior/purpose/capabilities, etc.
appId	Integer	true	-	ID of the application that requested this RPC.

7.29.3 Response

Note:

There is a difference in message type for WebSocket and D-Bus connection (see the table below).

Result	Description	Message type		Message Params	Notes
		WebSocket	D-Bus		
Success	SUCCESS: HMI recognizes the requested.template name and provides the capabilities of this template.	JSON response	Method return	displayCapabilities, buttonCapabilities, softButtonCapabilities, presetBankCapabilities, code: 0	See section 7.29.3.1.
Failure	UNsupported_RESOURCE: HMI does not support any or one display templates at all (response to UI.GetCapabilities contains an empty array of templatesAvailable).	JSON error message	Method return	code: 2	Applicable for this RPC result codes.
	IGNORED: The named template is already associated with the named appId application			code: 6	Please see Result Enumeration for all SDL-supported codes.

	INVALID_DATA: The requested template name is not valid.			code: 11	
	INVALID_ID appId is invalid (e.g. doesn't exist)			code: 13	
	GENERIC_ERROR 1) The unknown issue occurred or other codes are not applicable.			code: 22	

SDL 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.

7.29.3.1 Parameters

Param Name	Type	Mandatory	Additional	Description
displayCapabilities	Common.DisplayCapabilities	false	-	The type of display, supported text and image fields, formats of media clock and other that is described in section 7.29.3.2 .
buttonCapabilities	Common.ButtonCapabilities	false	Array = true Minsize = 1 Maxsize = 100	Capabilities of hardware buttons: the names of available buttons, information about whether notifications on short/long presses and up/down events are supported by HMI (see section 7.29.3.15).
softButtonCapabilities	Common.SoftButtonCapabilities	false	Array = true Minsize = 1 Maxsize = 100	Must be returned when the template supports on-screen SoftButtons. Capabilities of soft buttons: information about whether HMI supports referencing image and notifying on short/long presses and up/down events (see section 7.29.3.17).
presetBankCapabilities	Common.PresetBankCapabilities	false	-	Must be returned when the template supports custom on-screen presets. Information about whether HMI supports duplicating the hardware custom presets with onscreen buttons (see section 7.29.3.18).

7.29.3.2 DisplayCapabilities

Param Name	Type	Mandatory	Additional	Description
displayType	Common.DisplayType	true	-	The type of the display that is installed on HU. See section 7.29.3.5 DisplayType .

Param Name	Type	Mandatory	Additional	Description
textFields	Common.TextField	true	Array = true minsize = 0 maxsize = 100	A set of all fields that support displaying text data. If there are no textfields supported HMI must send the empty array. See section 7.29.3.6 TextField.
imageFields	Common.ImageField	false	Array = true minsize = 1 maxsize = 100	A set of all fields that support images. See section 7.29.3.7 ImageField
mediaClockFormats	Common.MediaClockFormat	true	Array = true minsize = 1 maxsize = 100	A set of all supported formats of the media clock. See section 7.29.3.11 MediaClockFormat.
imageCapabilities	Common.ImageType	false	Array = true minsize = 0 maxsize = 2	The array of supported image types (static and/or dynamic). The empty array should be returned if the platform does not support displaying images. See section 7.29.3.12 ImageType.
graphicSupported	Boolean	true	-	Must be: - 'true' if display's persistent screen supports referencing a static or dynamic image. - 'false' if not.
templatesAvailable	String	true	Array = true minsize = 0 maxsize = 100 maxlength = 100	A set of all predefined persistent display templates available on headunit. See section 7.29.3.17 PredefinedLayout for the variants of template names. Still, HMI is not limited with these values and is free to create the name(s) that fit the template behavior/purpose/capabilities, etc.
screenParams	Common.ScreenParams	false	-	A set of all parameters related to a prescribed screen area (e.g. for video / touch input). See section 7.29.3.11 ScreenParams.
numCustomPresetsAvailable	Integer	false	minvalue = 1 maxvalue = 100	The number of on-screen custom presets available (if any); otherwise omitted.

7.29.3.3 DisplayType Enumeration

Element name	Short Description
CID	Center Information Display. This display type provides a 2-line x 20 character "dot matrix" display.
TYPE2	TYPE II display. 1 line older radio head unit.
TYPE5	TYPE V display Old radio head unit.
NGN	Next Generation Navigation display.
GEN2_8_DMA	GEN-2, 8 inch display.
GEN2_6_DMA	GEN-2, 6 inch display.

Element name	Short Description
MFD3	3 inch GEN1.1 display
MFD4	4 inch GEN1.1 display
MFD5	5 inch GEN1.1 display
GEN3_8-INCH	GEN-3, 8 inch display.

7.29.3.4 *TextField*

Param Name	Type	Mandatory	Additional	Description
name	Common.TextField.Name	true	-	The name that identifies the field. See <i>TextFieldName</i> .
characterSet	Common.CharacterSet	true	-	The character set that is supported in this field. See <i>CharacterSet</i> .
width	Integer	true	minvalue="1" maxvalue="500"	The number of characters in one row of this field.
rows	Integer	true	minvalue="1" maxvalue="3"	The number of rows of this field.

7.29.3.5 *TextFieldName Enumeration*

Element name	Short Description
mainField1	The text that must be displayed in a single or upper display line. If this value is not set, the text of mainField1 must stay unchanged. If this text is empty "", the field must be cleared. Applies to <i>Show</i> , section 7.5 .
mainField2	The text that must be displayed on the second display line. If this text is not set, the text of mainField2 must stay unchanged. If this text is empty "", the field must be cleared. Applies to <i>Show</i> , section 7.5 .
mainField3	The text that must be displayed on the second "page" first display line. If this text is not set, the text of mainField3 must stay unchanged. If this text is empty "", the field must be cleared. Applies to <i>Show</i> , section 7.5 .

Element name	Short Description
mainField4	<p>The text that must be displayed on the second "page" second display line.</p> <p>If this text is not set, the text of mainField4 must stay unchanged.</p> <p>If this text is empty "", the field must be cleared.</p> <p>Applies to Show, section 7.5.</p>
statusBar	<p>The text is placed in the status bar area.</p> <p>Note: This relates to navigation displays</p> <p>If this parameter is omitted, the status bar text must remain unchanged.</p> <p>If this parameter is an empty string, the field must be cleared.</p> <p>If provided and the display has no status bar, this parameter must be ignored.</p> <p>Applies to Show, section 7.5.</p>
mediaClock	<p>Text value for MediaClock field. Shall arrive in the form as described in the MediaClockFormat enumeration</p> <p>If this text is set, any automatic media clock updates previously set with SetMediaClockTimer must be stopped.</p> <p>Applies to Show, section 7.5.</p>
mediaTrack	<p>The text that should be displayed in the track field. This field should be valid only for media applications on.</p> <p>If this text is not set, the text of mediaTrack must stay unchanged.</p> <p>If this text is empty "", the field must be cleared.</p> <p>Applies to Show, section 7.5.</p>

Element name	Short Description
alertText1	The text that must be displayed in the top field of the display during the Alert. Applies to Alert, section 7.4.
alertText2	The text that must be displayed in the bottom field of the display during the Alert. Applies to Alert, section 7.4.
alertText3	The optional third line of the alert text field. Applies to Alert, section 7.4.
scrollableMessageBody	The long form body of text that can include newlines and tabs. Applies to ScrollableMessage, section 7.17.
initialInteractionText	Must be displayed when the interaction begins. The text must be displayed on the first line of a multiline display, and must be centered. Applies to PerformInteraction, section 7.10.
navigationText1	The text that must be displayed on the first line of navigation text. Applies to ShowConstantTBT, section 12.3.
navigationText2	The text that must be displayed on the second line of navigation text. Applies to ShowConstantTBT, section 12.3.
ETA	Estimated Time of Arrival for navigation. Applies to ShowConstantTBT, section 12.3.
totalDistance	Total distance to destination for navigation. Applies to ShowConstantTBT, section 12.3.
navigationText	Navigation text for UpdateTurnList. Applies to Turn, section 12.4.

Element name	Short Description
audioPassThruDisplayText1	The first line of text that must be displayed during audio capture. Applies to PerformAudioPassThru, section 7.20.
audioPassThruDisplayText2	The second line of text that must be displayed during audio capture. Applies to PerformAudioPassThru, section 7.20.
sliderHeader	The text that must be displayed on the header of slider. Applies to Slider, section 7.18.
sliderFooter	The text that must be displayed on the footer of slider. Applies to Slider, section 7.18.
notificationText	The text that must be displayed to notify the User on some event. Applies to ShowNotification, section 7.31.
menuName	Primary text for Choice. Applies to PerformInteraction, section 7.12.
secondaryText	Secondary text for Choice. Applies to PerformInteraction, section 7.12.
tertiaryText	Tertiary text for Choice. Applies to PerformInteraction, section 7.12.
timeToDestination	The line to display the time to destination. Applies to ShowConstantTBT, section 12.3.
turnText	Currently not used.

7.29.3.6 CharacterSet

Element name	Value	Short Description
TYPE2SET	0	
TYPE5SET	1	
CID1SET	2	
CID2SET	3	

7.29.3.7 ImageField Structure

Param Name	Type	Mandatory	Additional	Description
name	Common.ImageFileDialogName	true	-	The name that identifies the field. See section 7.32.3.6 ImageFieldName.
imageTypeSupported	Common.FileType	false	array = true minsize = 1 maxsize = 100	The image types that are supported in this field. See section 7.32.3.7 FileType.
imageResolution	Common.ImageResolution	false	-	The image resolution of this field. See section 7.32.3.8.

7.29.3.8 ImageFieldName Enumeration

Element name	Value	Short Description
softButtonImage	0	The image field for SoftButton
choiceImage	1	The first image field for Choice
choiceSecondaryImage	2	The secondary image field for Choice
vrHelpItem	3	The image field for vrHelpItem
turnIcon	4	The image field for Turn
menuIcon	5	The image field for the menu icon in SetGlobalProperties
cmdIcon	6	The image field for AddCommand
appIcon	7	The image field for the app icon (set by setAppIcon)
graphic	8	The image field for Show
showConstantTBTIcon	9	The primary image field for ShowConstantTBT
showConstantTBTNextTurnIcon	10	The secondary image field for ShowConstantTBT
locationImage	11	The optional image of a destination / location

7.29.3.9 FileType Enumeration

Element name	Value	Short Description
GRAPHIC_BMP	0	
GRAPHIC_JPEG	1	
GRAPHIC_PNG	2	
AUDIO_WAVE	3	
AUDIO_MP3	4	
AUDIO_AAC	5	
BINARY	6	
JSON	7	

7.29.3.10 ImageResolution Structure

Param Name	Type	Mandatory	Additional	Description
resolutionWidth	Integer	true	minvalue = 1 maxvalue = 10000	The image resolution width.
resolutionHeight	Integer	true	minvalue = 1 maxvalue = 10000	The image resolution height.

7.29.3.11 MediaClockFormat Enumeration

Element name	Short Description
CLOCK1	<pre>minutesFieldWidth = 2; minutesFieldMax = 19; secondsFieldWidth = 2; secondsFieldMax = 99; maxHours = 19; maxMinutes = 59; maxSeconds = 59; Is used for Type II, NGN and CID head units.</pre>
CLOCK2	<pre>minutesFieldWidth = 3; minutesFieldMax = 199; secondsFieldWidth = 2; secondsFieldMax = 99; maxHours = 59; maxMinutes = 59; maxSeconds = 59; Is used for Type V head units.</pre>
CLOCK3	<pre>minutesFieldWidth = 2; minutesFieldMax = 59; secondsFieldWidth = 2; secondsFieldMax = 59; maxHours = 9; maxMinutes = 59; maxSeconds = 59; Is used for GEN1.1 (i.e. MFD3/4/5) head units.</pre>

Element name	Short Description
CLOCKTEXT1	5 characters possible Format: 1 sp c : sp c c 1 sp : digit "1" or space c : character out of following character set: sp 0-9 [letters] : sp : colon or space Is used for Type II head unit
CLOCKTEXT2	5 chars possible Format: 1 sp c : sp c c 1 sp : digit "1" or space c : character out of following character set: sp 0-9 [letters] : sp : colon or space Is used for CID and NGN head unit.
CLOCKTEXT3	6 chars possible Format: 1 sp c c : sp c c 1 sp : digit "1" or space c : character out of following character set: sp 0-9 [letters] : sp : colon or space Is used for Type V head unit.
CLOCKTEXT4	6 chars possible Format: c : sp c c : c c : sp : colon or space c : character out of following character set: sp 0-9 [letters]. Is used for GEN1.1 (i.e. MFD3/4/5) head units.

7.29.3.12 ImageType Enumeration

Element name	Short Description
STATIC	Static image. The image that is sent as the binary or hex code within the request.
DYNAMIC	Dynamic image. The image that is stored on HMI and just a link to it is further used within requests.

7.29.3.13 ScreenParams Structure

Param Name	Type	Mandatory	Description
resolution	Common.ImageResolution	true	The resolution of the prescribed screen area. See section 7.32.3.8 ImageResolution.
touchEventAvailable	Common.TouchEventCapabilities	false	Types of screen touch events available in screen area. See section 7.32.3.12.

7.29.3.14 TouchEventCapabilities Structure

Param Name	Type	Mandatory	Description
pressAvailable	Boolean	true	
multiTouchAvailable	Boolean	true	
doublePressAvailable	Boolean	true	

7.29.3.15 ButtonCapabilities

Param Name	Type	Mandatory	Description
name	Common.ButtonName	true	The name of supported/existing hardware button. See section 7.32.3.14 ButtonName
shortPressAvailable	Boolean	true	Must be 'true' if the button supports short press mode. If 'true' HMI must notify SDL with OnButtonPress (SHORT) every time this button is pressed short. See section 8.3 OnButtonPress
longPressAvailable	Boolean	true	Must be 'true' if the button supports long press mode. If 'true' HMI must notify SDL with OnButtonPress (LONG) every time this button is pressed long. See section 8.3 OnButtonPress
upDownAvailable	Boolean	true	Must be 'true' if tracking the events of button being depressed/released is supported. If 'true' HMI must notify SDL with OnButtonEvent of DOWN/UP on every button depress/release. See section 8.4 OnButtonEvent

7.29.3.16 ButtonName

Element name	Value	Short Description
OK	0	Represents the button usually labeled "OK". A typical use of this button is for the User to press it to make a selection.
SEEKLEFT	1	Represents the seek-left button. A typical use of this button is for the user to scroll to the left through menu choices, one menu item per press.
SEEKRIGHT	2	Represents the seek-right button. A typical use of this button is for the user to scroll to the right through menu choices one menu item per press.
TUNEUP	3	Represents a turn of the tuner knob in the clockwise direction one tick.
TUNEDOWN	4	Represents a turn of the tuner knob in the counter-clockwise direction one tick.

Element name	Value	Short Description
PRESET_0	5	Represents the preset 0 button.
PRESET_1	6	Represents the preset 1 button.
PRESET_2	7	Represents the preset 2 button.
PRESET_3	8	Represents the preset 3 button.
PRESET_4	9	Represents the preset 4 button.
PRESET_5	10	Represents the preset 5 button.
PRESET_6	11	Represents the preset 6 button.
PRESET_7	12	Represents the preset 7 button.
PRESET_8	13	Represents the preset 8 button.
PRESET_9	14	Represents the preset 9 button.
CUSTOM_BUTTON	15	Represents any of onscreen buttons requested by SDL.
SEARCH	16	Represents the 'Search' button on the touchscreen keyboard.
PLAY_PAUSE	17	Represents Play/Pause button functionality (e.g. Play and Pause audio source)

7.29.3.17 SoftButtonCapabilities Structure

Param Name	Type	Mandatory	Description
shortPressAvailable	Boolean	true	Must be - 'true' if soft buttons support a short press - 'false' if not. See section 8.3 OnButtonPress.
longPressAvailable	Boolean	true	Must be - 'true' if soft buttons support a LONG press - 'false' if not. See section 8.3 OnButtonPress.

Param Name	Type	Mandatory	Description
upDownAvailable	Boolean	true	Must be - 'true' if soft buttons support "button down" and "button up". - 'false' if not. See section 8.4 OnButtonEvent
imageSupported	Boolean	true	Must be - 'true' if soft buttons support referencing image - 'false' if not.

7.29.3.18 PresetBankCapabilities

Param Name	Type	Mandatory	Description
onScreenPresetsAvailable	Boolean	true	Must be 'true' if HMI supports duplicating the hardware custom presets with onscreen buttons.

7.29.3.19 PredefinedLayout

Element name	Short Description
DEFAULT	Default media / non-media screen. Can be set as a root screen.
MEDIA	Default Media screen. Can be set as a root screen.
NON-MEDIA	Default Non-media screen. Can be set as a root screen.

Element name	Short Description
ONSCREEN_PRESETS	Custom root media screen containing app-defined onscreen presets. Can be set as a root screen.
NAV_FULLSCREEN_MAP	Custom root template screen containing full screen map with navigation controls. Can be set as a root screen.
NAV_LIST	Custom root template screen containing video represented list. Can be set as a root screen.
NAV_KEYBOARD	Custom root template screen containing video represented keyboard. Can be set as a root screen.
GRAPHIC_WITH_TEXT	Custom root template screen containing half-screen graphic with lines of text. Can be set as a root screen.

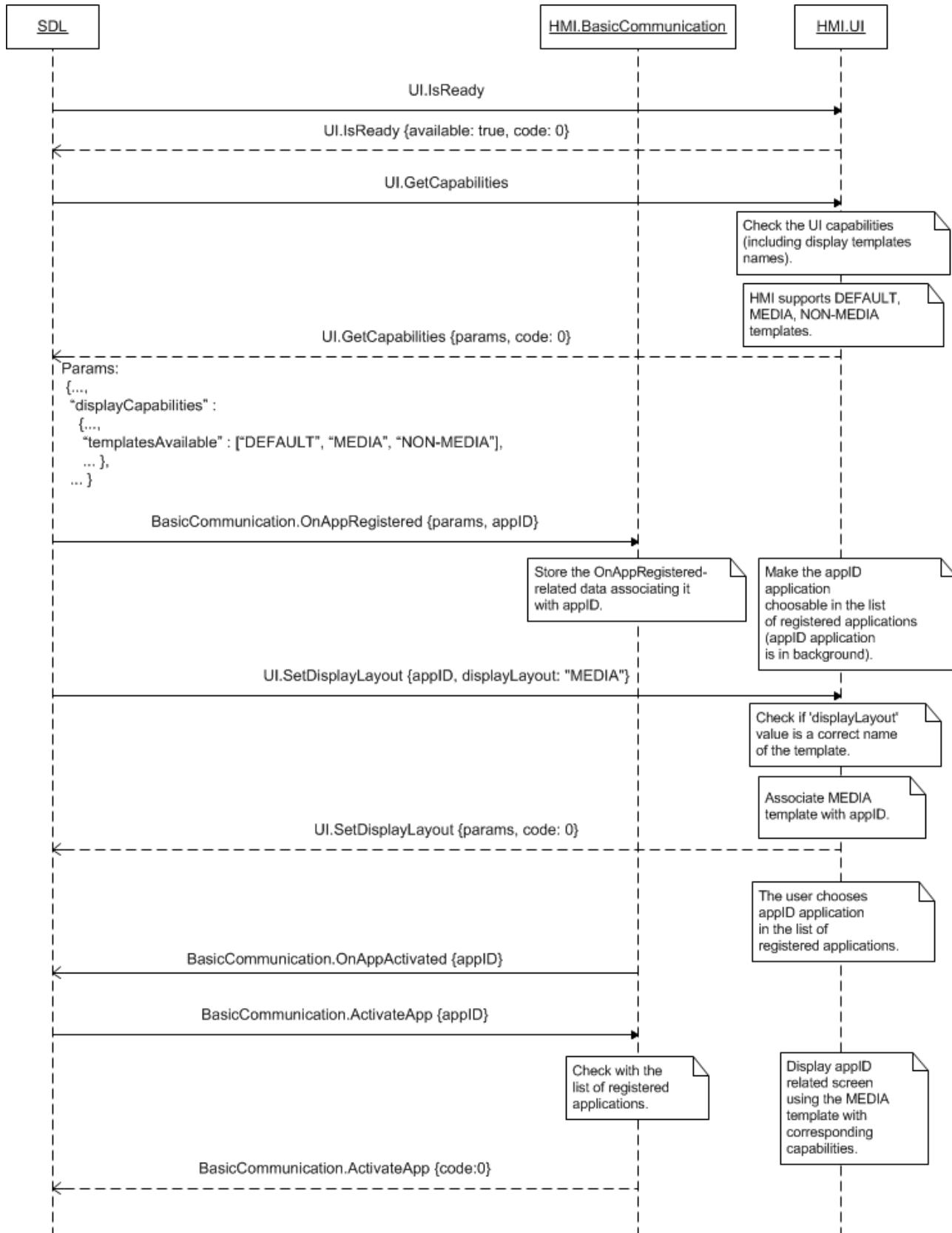
Element name	Short Description
TEXT_WITH_GRAPHIC	Custom root template screen containing lines of text with half-screen graphic. Can be set as a root screen.
TILES_ONLY	Custom root template screen containing only tiled SoftButtons . Can be set as a root screen.
TEXTBUTTONS_ONLY	Custom root template screen containing only text SoftButtons . Can be set as a root screen.
GRAPHIC_WITH_TILES	Custom root template screen containing half-screen graphic with tiled SoftButtons . Can be set as a root screen.
TILES_WITH_GRAPHIC	Custom root template screen containing tiled SoftButtons with half-screen graphic. Can be set as a root screen.

Element name	Short Description
GRAPHIC_WITH_TEXT_AND_SOFTBUTTONS	Custom root template screen containing half-screen graphic with text and SoftButtons . Can be set as a root screen.
TEXT_AND_SOFTBUTTONS_WITH_GRAPHIC	Custom root template screen containing text and SoftButtons with half-screen graphic. Can be set as a root screen.
GRAPHIC_WITH_TEXTBUTTONS	Custom root template screen containing half-screen graphic with text only SoftButtons . Can be set as a root screen.
TEXTBUTTONS_WITH_GRAPHIC	Custom root template screen containing text only SoftButtons with half-screen graphic. Can be set as a root screen.

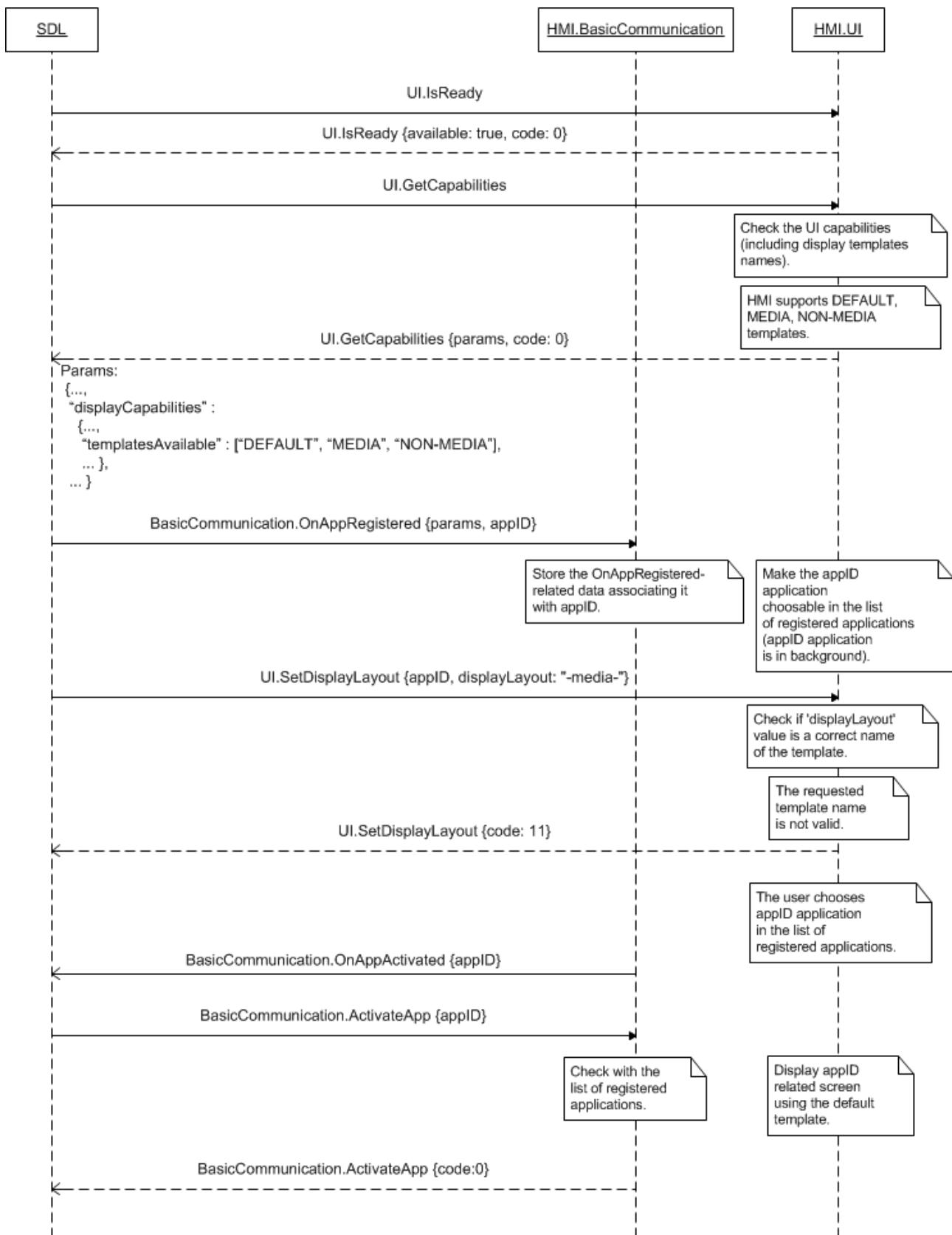
Element name	Short Description
LARGE_GRAPHIC_WITH_SOFTBUTTONS	Custom root template screen containing a large graphic and SoftButtons . Can be set as a root screen.
DOUBLE_GRAPHIC_WITH_SOFTBUTTONS	Custom root template screen containing two graphics and SoftButtons . Can be set as a root screen.
LARGE_GRAPHIC_ONLY	Custom root template screen containing only a large graphic. Can be set as a root screen.

7.29.4 Sequence Diagrams

7.29.4.1 SetDisplayLayout (successful) with preceding UI.GetCapabilities



7.29.4.2 SetDisplayLayout (INVALID_DATA) with preceding UI.GetCapabilities



7.29.5 JSON Messages Examples

7.29.5.1 Request

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

7.29.5.2 Response

```
{  
    "id" : 47,  
    "jsonrpc" : "2.0",  
    "result" :  
    {  
        "displayCapabilities" :  
        {  
            "displayType" :  
GEN2_8_DMA,  
            "textFields" :  
[mainField1, mainField2, alertText1,  
  
            alertText2, alertText3,  
scrollableMessageBody,  
            initialInteractionText,  
navigationText1,  
            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"
}
}

```

7.29.5.3 Error message

```

{
  "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"
    }
  }
}

```

8 Buttons Component Description

8.1 GetCapabilities

8.1.1 Description

Type:	Function
Sender:	SDL
Purpose:	Get the capabilities of hardware buttons.

On startup after getting OnReady notification from HMI, SDL sends the Buttons.GetCapabilities RPC to obtain the buttons capabilities.

SDL Note: In case SDL didn't get Button capabilities from HMI, it will use the default ones from the file hmi_capabilities.json. The path to the file is defined in smartDeviceLink.ini as HMICapabilities parameter. See also [15.3.2.4 Buttons section of hmi_capabilities.json](#)

8.1.2 Request

8.1.2.1 Behavior

HMI must:

- 1) Check the capabilities of supported hardware buttons and provide the following information about each of them:

- Name (from the ButtonName enumeration).

Note:

CUSTOM_BUTTON name and the corresponding capabilities must be provided for the case when HMI has the ability to draw custom buttons onscreen.

- Whether the short/long presses (see Chapter 1 “Abbreviations and Definitions”) are supported and distinguished by HMI. (If supported, HMI must notify SDL with OnButtonPress of SHORT/LONG on every button press).
- Whether tracking the events of button being depressed and released are supported. (If supported, HMI must notify SDL with OnButtonEvent of DOWN/UP on every button depress/release).

Note:

For the mobile application better performance on HMI, HMI is recommended to provide the ability of reporting on depress/release events and at least ‘short’ press mode for its supported hardware and custom onscreen buttons.

- 2) Check whether the ability to duplicate the hardware preset buttons with onscreen buttons is supported.
- 3) Respond correspondingly to results of the check.

8.1.3 Response

Note:

There is a difference in message type for WebSocket and D-Bus connection (see the table below).

Result	Description	Message type		Message Params	Notes
		WebSocket	D-Bus		
Success	SUCCESS: HMI provides the buttons capabilities.	JSON response	Method return	capabilities, presetBankCap abilities, code: 0	
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 for this RPC result codes. Please see Result Enumeration for all SDL-supported codes.
	DATA_NOT_AVAILABLE: The information about buttons capabilities cannot be provided.			Code: 9	
	GENERIC_ERROR: The unknown issue occurred or other codes are not applicable.			code: 22	

8.1.3.1 Parameters

Param name	Type	Mandatory	Additional	Description
capabilities	Common.ButtonCapabilities	true	array = true minsize = 1 maxsize = 100	Each element of the array defines the button name and whether short/long presses and down/up events are supported. See ButtonCapabilities
presetBank Capabilities	Common.PresetBankCapabilities	false	-	Should be returned if the platform supports duplicating the hardware presets with onscreen buttons. See PresetBankCapabilities

8.1.3.2 ButtonCapabilities

Param Name	Type	Mandatory	Description
name	Common.ButtonName	true	The name of supported/existing hardware button. See ButtonName
shortPressAvailable	Boolean	true	Must be 'true' if the button supports short press mode. If 'true' HMI must notify SDL with OnButtonPress (SHORT) every time this button is pressed short. See ButtonPressMode
longPressAvailable	Boolean	true	Must be 'true' if the button supports long press mode. If 'true' HMI must notify SDL with OnButtonPress (LONG) every time this button is pressed long. See ButtonPressMode
upDownAvailable	Boolean	true	Must be 'true' if tracking the events of button being depressed/released is supported. If 'true' HMI must notify SDL with OnButtonEvent of DOWN/UP on every button depress/release. See ButtonEventMode

8.1.3.3 ButtonName

Element name	Value	Short Description
OK	0	Represents the button usually labeled "OK". A typical use of this button is for the User to press it to make a selection.
SEEKLEFT	1	Represents the seek-left button. A typical use of this button is for the user to scroll to the left through menu choices, one menu item per press.
SEEKRIGHT	2	Represents the seek-right button. A typical use of this button is for the user to scroll to the right through menu choices one menu item per press.
TUNEUP	3	Represents a turn of the tuner knob in the clockwise direction one tick.

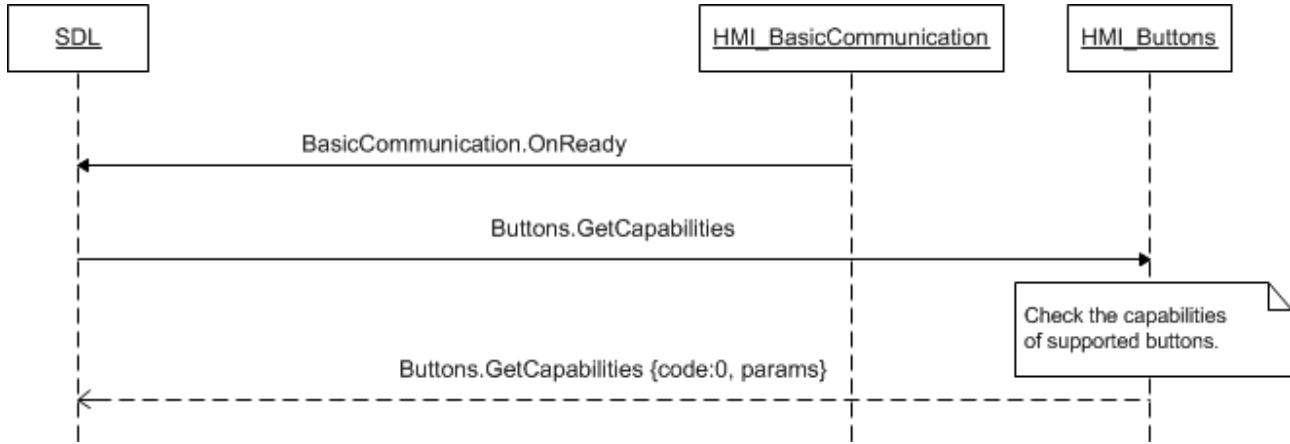
Element name	Value	Short Description
TUNEDOWN	4	Represents a turn of the tuner knob in the counter-clockwise direction one tick.
PRESET_0	5	Represents the preset 0 button.
PRESET_1	6	Represents the preset 1 button.
PRESET_2	7	Represents the preset 2 button.
PRESET_3	8	Represents the preset 3 button.
PRESET_4	9	Represents the preset 4 button.
PRESET_5	10	Represents the preset 5 button.
PRESET_6	11	Represents the preset 6 button.
PRESET_7	12	Represents the preset 7 button.
PRESET_8	13	Represents the preset 8 button.
PRESET_9	14	Represents the preset 9 button.
CUSTOM_BUTTON	15	Represents any of onscreen buttons requested by SDL.
SEARCH	16	Represents the 'Search' button on the touchscreen keyboard.
PLAY_PAUSE	17	Represents Play/Pause button functionality (e.g. Play and Pause audio source)

8.1.3.4 PresetBankCapabilities

Param Name	Type	Mandatory	Description
onScreenPresetsAvailable	Boolean	true	Must be 'true' if HMI supports duplicating the hardware custom presets with onscreen buttons.

8.1.4 Sequence Diagrams

8.1.4.1 GetCapabilities on system startup



8.1.5 JSON Messages Examples

8.1.5.1 Request

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

8.1.5.2 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"
}
}

```

8.1.5.3 Error message

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

8.2 OnButtonEvent

8.2.1 Description

Type:	Notification
Sender:	HMI
Purpose:	Inform about the event of depress/relese has occurred for the button.

Once HMI confirmed it supports reporting on events of button press/release for the named buttons (via response to Buttons.GetCapabilities), SDL expects to be notified about such events via OnButtonEvent.

Important: OnButtonEvent notification is sent by HMI only for a certain buttonNames on which an application is subscribed to (see [8.4 OnButtonSubscription](#)).

HMI must:

Send the following information upon every event of press or release occurred for each of subscribed buttons:

- 1) The name of the button (from ButtonName enumeration) the event has occurred for.

The event occurred:

- BUTTONDOWN – when the User has pressed the button
 - BUTTONUP – when the User has released the button.
- 2) The ID of the custom button: for the case of CUSTOM_BUTTON name only.
 - 3) The appId in case the ButtonName is CUSTOM_BUTTON (if not sent, SDL will ignore the notification and will not transfer it to mobile app).

Note:

The value of custom button ID is provided by SDL within softButton structure (softButtonID parameter) via one of the following RPCs: UI.Alert, UI.ScrollableMessage, UI.Show, Navi.ShowConstantTBT etc.

Note: SDL ignores all invalid notifications which come from HMI (invalid JSON, invalid data types/bounds etc)

8.2.1.1 Parameters

Param name	Type	Mandatory	Additional	Description
name	Common.ButtonName	true	–	The name of the button involved into event. See ButtonName.
mode	Common.ButtonEventMode	true	–	Indicates whether this is an UP or DOWN event See ButtonEventMode.
customButtonID	Integer	false	minvalue = 0 maxvalue = 65536	This ID must be provided in case the event has occurred for the CUSTOM_BUTTON only. The value is previously provided by SDL within softButton structure via one of the following RPCs: UI.Alert, UI.ScrollableMessage, UI.Show etc
appId	Integer	false	–	In case the ButtonName is CUSTOM_BUTTON, HMI must include appId parameters to OnButtonPress notification sent to SDL. Otherwise, if appId is not sent together with CUSTOM_BUTTON, this notification will be ignored by SDL.

8.1.3.3 ButtonName

Element name	Value	Short Description
OK	0	Represents the button usually labeled "OK". A typical use of this button is

Element name	Value	Short Description
		for the User to press it to make a selection.
SEEKLEFT	1	Represents the seek-left button. A typical use of this button is for the user to scroll to the left through menu choices, one menu item per press.
SEEKRIGHT	2	Represents the seek-right button. A typical use of this button is for the user to scroll to the right through menu choices one menu item per press.
TUNEUP	3	Represents a turn of the tuner knob in the clockwise direction one tick.
TUNEDOWN	4	Represents a turn of the tuner knob in the counter-clockwise direction one tick.
PRESET_0	5	Represents the preset 0 button.
PRESET_1	6	Represents the preset 1 button.
PRESET_2	7	Represents the preset 2 button.
PRESET_3	8	Represents the preset 3 button.
PRESET_4	9	Represents the preset 4 button.
PRESET_5	10	Represents the preset 5 button.
PRESET_6	11	Represents the preset 6 button.
PRESET_7	12	Represents the preset 7 button.
PRESET_8	13	Represents the preset 8 button.
PRESET_9	14	Represents the preset 9 button.
CUSTOM_BUTTON	15	Represents any of onscreen buttons requested by SDL.
SEARCH	16	Represents the 'Search' button on the touchscreen keyboard.
PLAY_PAUSE	17	Represents Play/Pause button functionality (e.g. Play and Pause audio source)

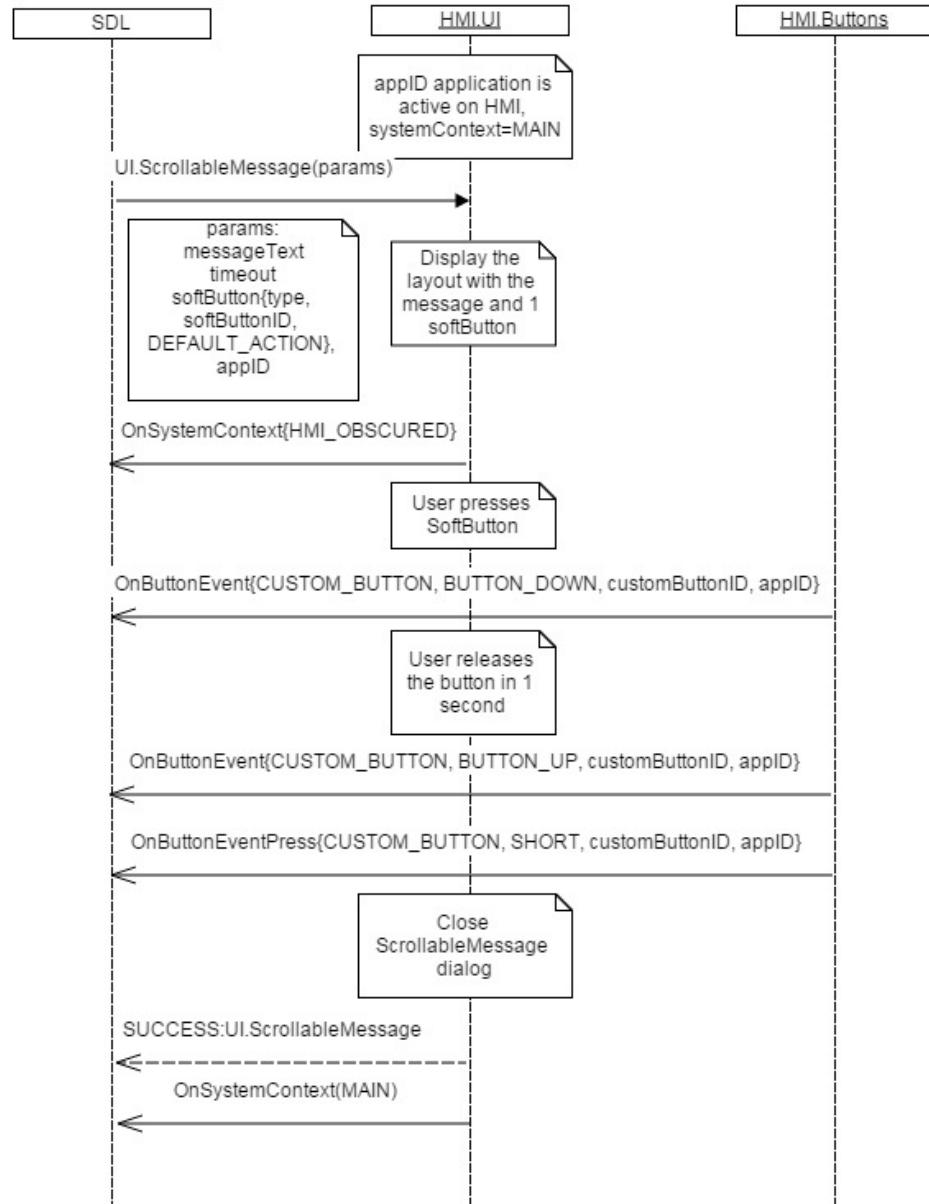
8.2.1.2 ButtonEventMode

Element name	Value	Short Description
BUTTONDOWN	0	The button has been pressed
BUTTONUP	1	The button has been

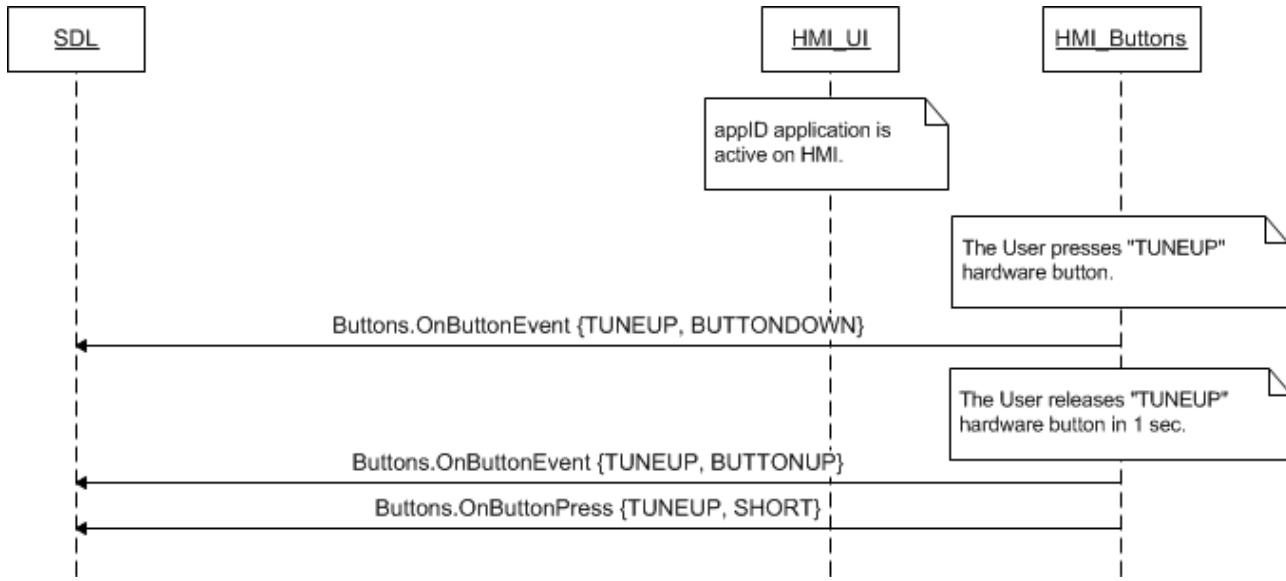
Element name	Value	Short Description
		released

8.2.2 Sequence Diagrams

8.2.2.1 OnButtonEvent for CUSTOM_BUTTON being pressed and released



8.2.2.1 OnButtonEvent for hardware button being pressed and released



8.2.3 JSON Messages Examples

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

8.3 OnButtonPress

8.3.1 Description

Type:	Notification
Sender:	HMI
Purpose:	Inform about the press mode detected for the button.

Once HMI confirmed it supports short and/or long press modes for the named buttons (via response to `Buttons.GetCapabilities`), SDL expects to be notified with the information of the press mode occurred via `OnButtonPress` in case of application subscription.

Important: `OnButtonPress` notification is sent by HMI only for a certain buttonNames on which an application is subscribed to (see [8.4 OnButtonSubscription](#)).

HMI must:

Send the following information upon every long or short press detected for each of subscribed buttons:

- 1) The name of the button (from `ButtonName` enumeration) being pressed.

- 2) The press mode detected:
 - SHORT – when the button is pressed and is being held during less than HMI-defined time threshold (e.g. during less than 2 sec.) and then released
 - LONG – when the button is pressed and is being held during more than HMI-defined time threshold (e.g. during more than 2 sec.) and then released
- 3) The ID of the custom button: for the case of CUSTOM_BUTTON name only.
- 4) The appId in case the ButtonName is CUSTOM_BUTTON (if not sent, SDL will ignore the notification and will NOT transfer it to mobile app).

Note:

The value of custom button ID is provided by SDL within softButton structure (softButtonID parameter) via one of the following RPCs: UI.Alert, UI.ScrollableMessage, UI.Show, Navi.ShowConstantTBT etc

Note: SDL ignores all invalid notifications which come from HMI (invalid JSON, invalid data types/bounds etc)

Important note:

It is expected that the sequence of OnButtonEvent and OnButtonPress notifications will depend on the button press mode detected:

- 1) *The button is pressed SHORT (is held during less than HMI-defined time threshold):*
 - OnButtonEvent (BUTTONDOWN)
 - OnButtonEvent (BUTTONUP)
 - OnButtonPress (SHORT)
- 2) *The button is pressed LONG (is held during more than HMI-defined time threshold):*
 - OnButtonEvent (BUTTONDOWN)
 - OnButtonPress (LONG)
 - OnButtonEvent (BUTTONUP)

Note:

In case the button supports SHORT press mode only, HMI must send OnButtonPress (SHORT) independently of the time this button is being held for.

8.3.1.1 Parameters

Param name	Type	Mandatory	Additional	Description
name	Common.ButtonName	true	–	The name of the button the press mode is detected for. See ButtonName.
mode	Common.ButtonPressMode	true	–	Indicates whether this is a LONG or SHORT button press mode See ButtonPressMode.
customButtonID	Integer	false	minvalue = 0 maxvalue = 65536	This ID must be provided in case the event has occurred for the CUSTOM_BUTTON only.

Param name	Type	Mandatory	Additional	Description
appID	Integer	false	-	In case the ButtonName is CUSTOM_BUTTON, HMI must include appID parameters to OnButtonPress notification sent to SDL. Otherwise, if appID is not sent together with CUSTOM_BUTTON, this notification will be ignored by SDL.

8.1.3.3 *ButtonName*

Element name	Value	Short Description
OK	0	Represents the button usually labeled "OK". A typical use of this button is for the User to press it to make a selection.
SEEKLEFT	1	Represents the seek-left button. A typical use of this button is for the user to scroll to the left through menu choices, one menu item per press.
SEEKRIGHT	2	Represents the seek-right button. A typical use of this button is for the user to scroll to the right through menu choices one menu item per press.
TUNEUP	3	Represents a turn of the tuner knob in the clockwise direction one tick.
TUNEDOWN	4	Represents a turn of the tuner knob in the counter-clockwise direction one tick.
PRESET_0	5	Represents the preset 0 button.
PRESET_1	6	Represents the preset 1 button.
PRESET_2	7	Represents the preset 2 button.
PRESET_3	8	Represents the preset 3 button.
PRESET_4	9	Represents the preset 4 button.
PRESET_5	10	Represents the preset 5 button.
PRESET_6	11	Represents the preset 6 button.
PRESET_7	12	Represents the preset 7 button.
PRESET_8	13	Represents the preset 8 button.
PRESET_9	14	Represents the preset 9 button.
CUSTOM_BUTTON	15	Represents any of

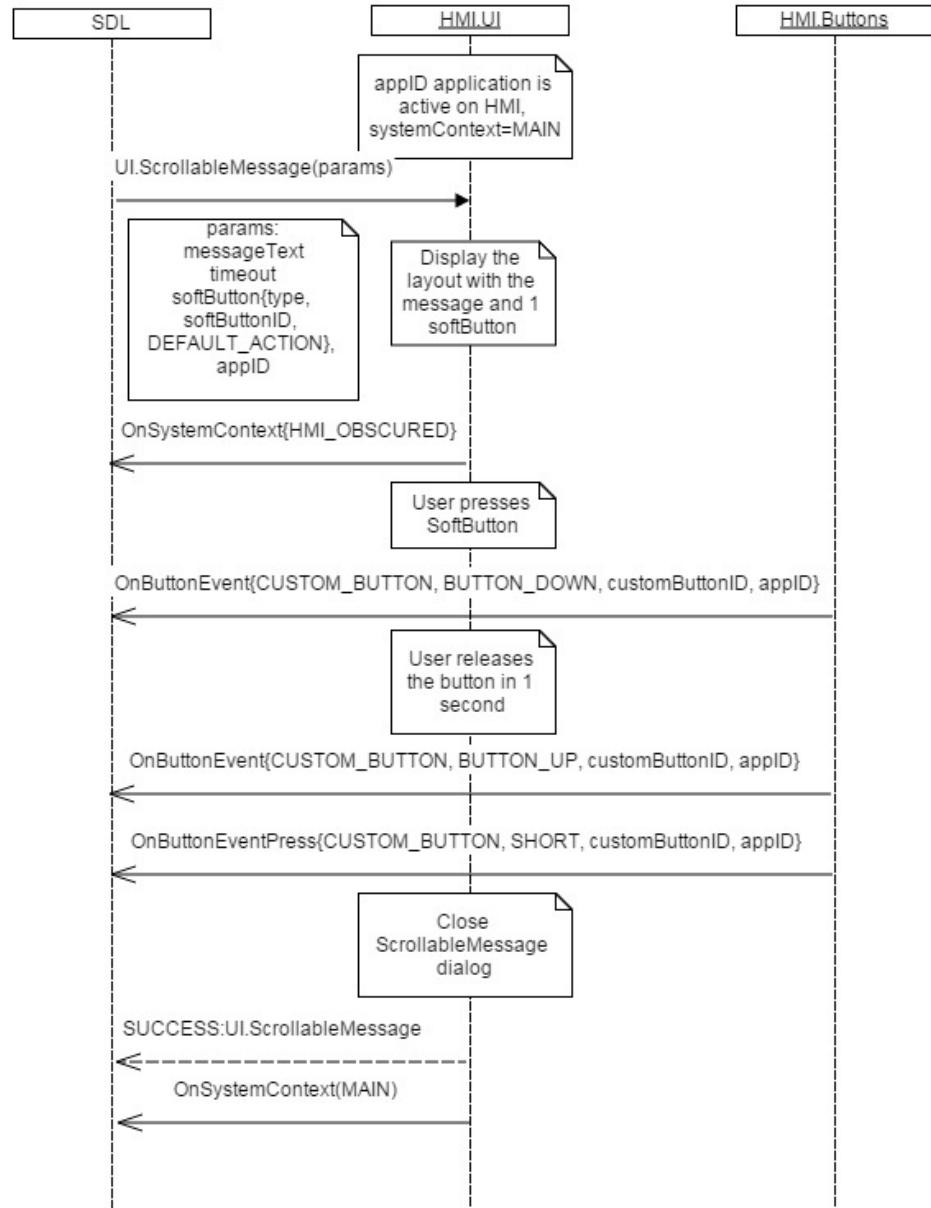
Element name	Value	Short Description
		onscreen buttons requested by SDL.
SEARCH	16	Represents the 'Search' button on the touchscreen keyboard.
PLAY_PAUSE	17	Represents Play/Pause button functionality (e.g. Play and Pause audio source)

8.3.1.2 ButtonPressMode

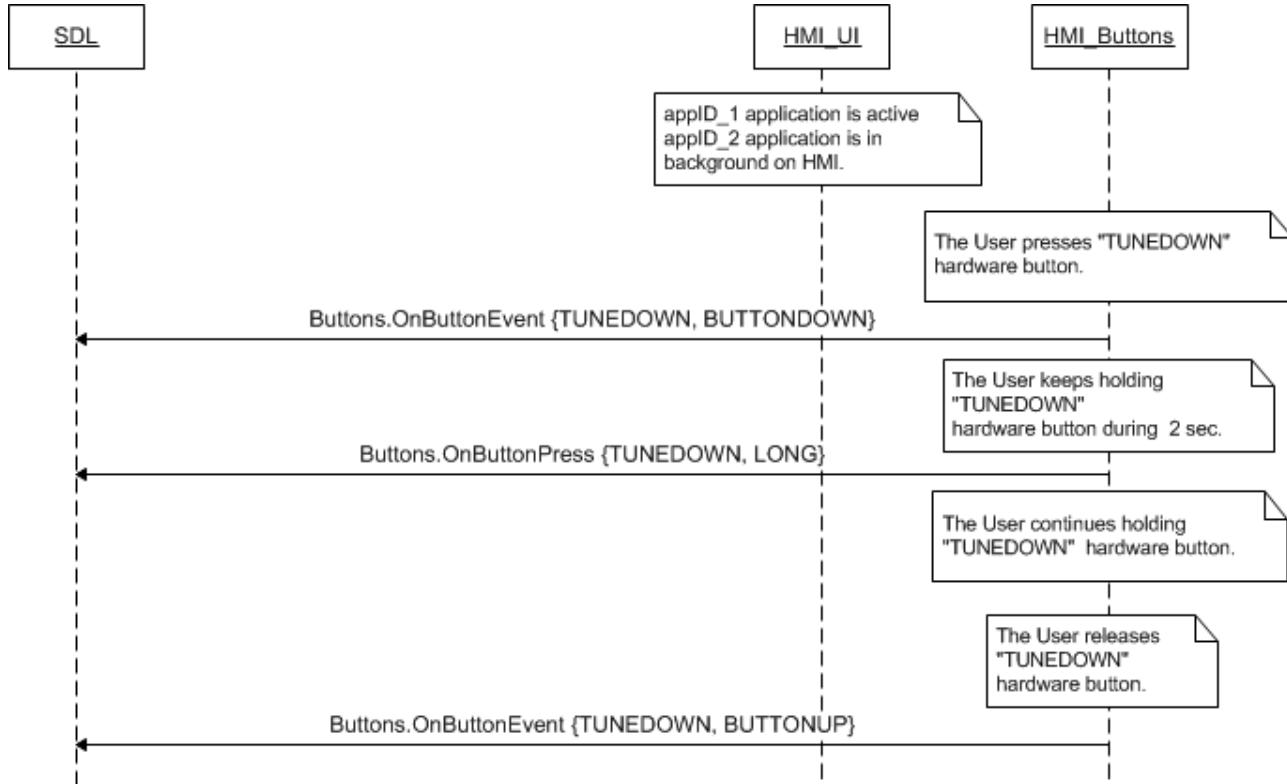
Element name	Value	Short Description
SHORT	0	Short-time button press: the button is being held during less than HMI-defined time threshold (e.g. during less than 2 sec.) and then released
LONG	1	Long-time button press: the button is being held during more than HMI-defined time threshold (e.g. during more than 2 sec.) and then released

8.3.2 Sequence Diagrams

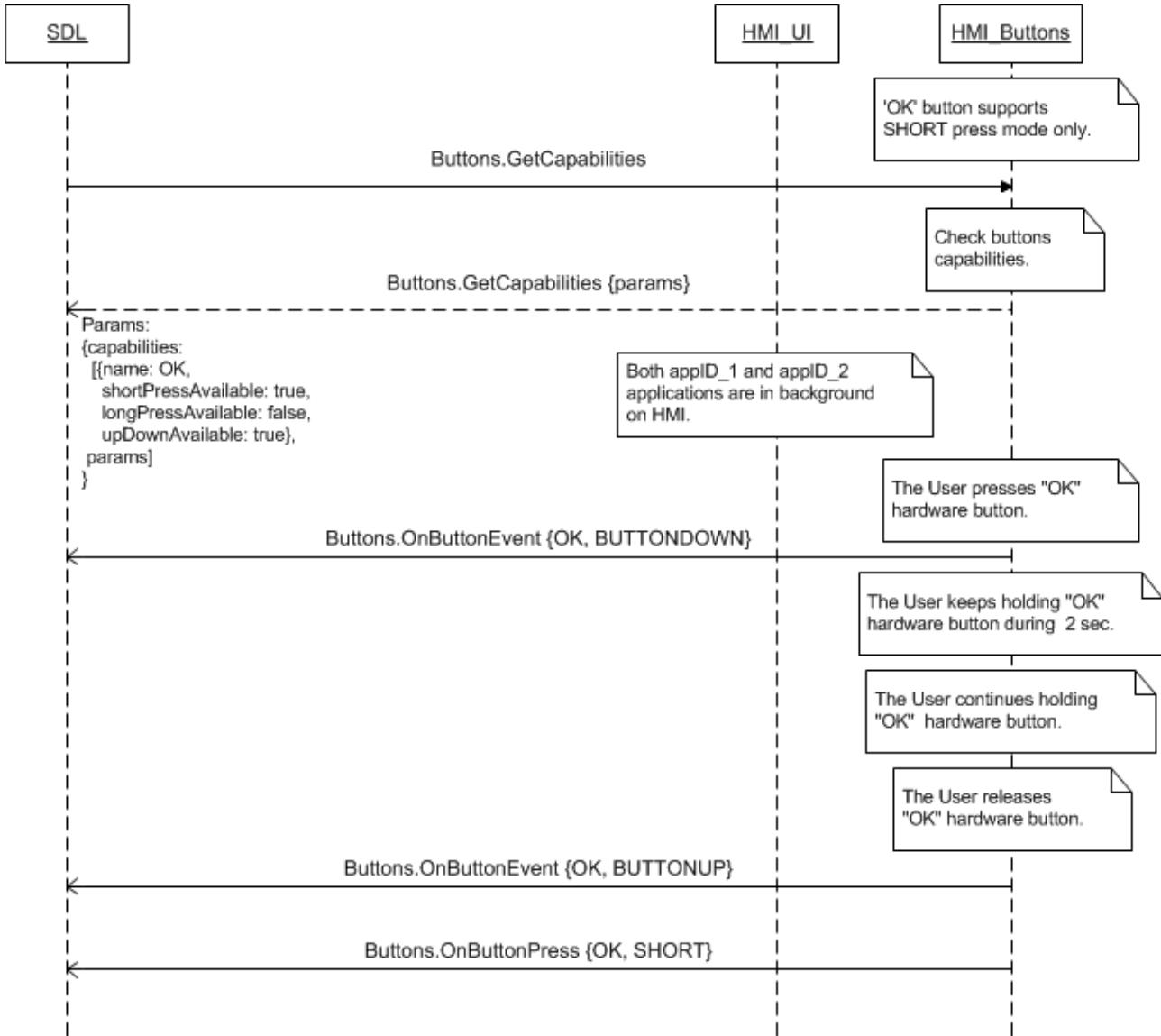
8.3.2.1 OnButtonPress (SHORT) for CUSTOM_BUTTON



8.3.2.2 *OnButtonPress (LONG) for hardware button*



8.3.2.2 OnButtonPress for hardware button that supports SHORT press mode only



8.3.3 JSON Messages Examples

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

8.4 OnButtonSubscription

8.4.1 Description

Type:	Notification
Sender:	SDL->HMI

Purpose:	To notify about button subscription state is changed for the named application
-----------------	--

HMI must:

- 1) HMI is expected to display on the screen layout only the duplicates of the HW buttons that the app is subscribed to.
- 2) In case SDL sends OnAppUnregistered to HMI (after the application has unregistered or as a result of disconnect), HMI must clean all the subscriptions related to this app.

Note: SDL subscribes each application to CUSTOM_BUTTON by default after each application registration. If CUSTOM_BUTTON is not supported by HMI, SDL must not send subscription notification.

8.4.1.1 Parameters

Param name	Type	Mandatory	Additional	Description
name	Common.ButtonName	true	-	The name of the button the press mode is detected for. See ButtonName.
isSubscribed	Boolean	true		Defines whether the named button has status of 'subscribed' or 'unsubscribed': If "true" - the named button is subscribed. If "false" - the named button is unsubscribed.
appID	Integer	true	-	The ID of application that relates to this button-subscription status change

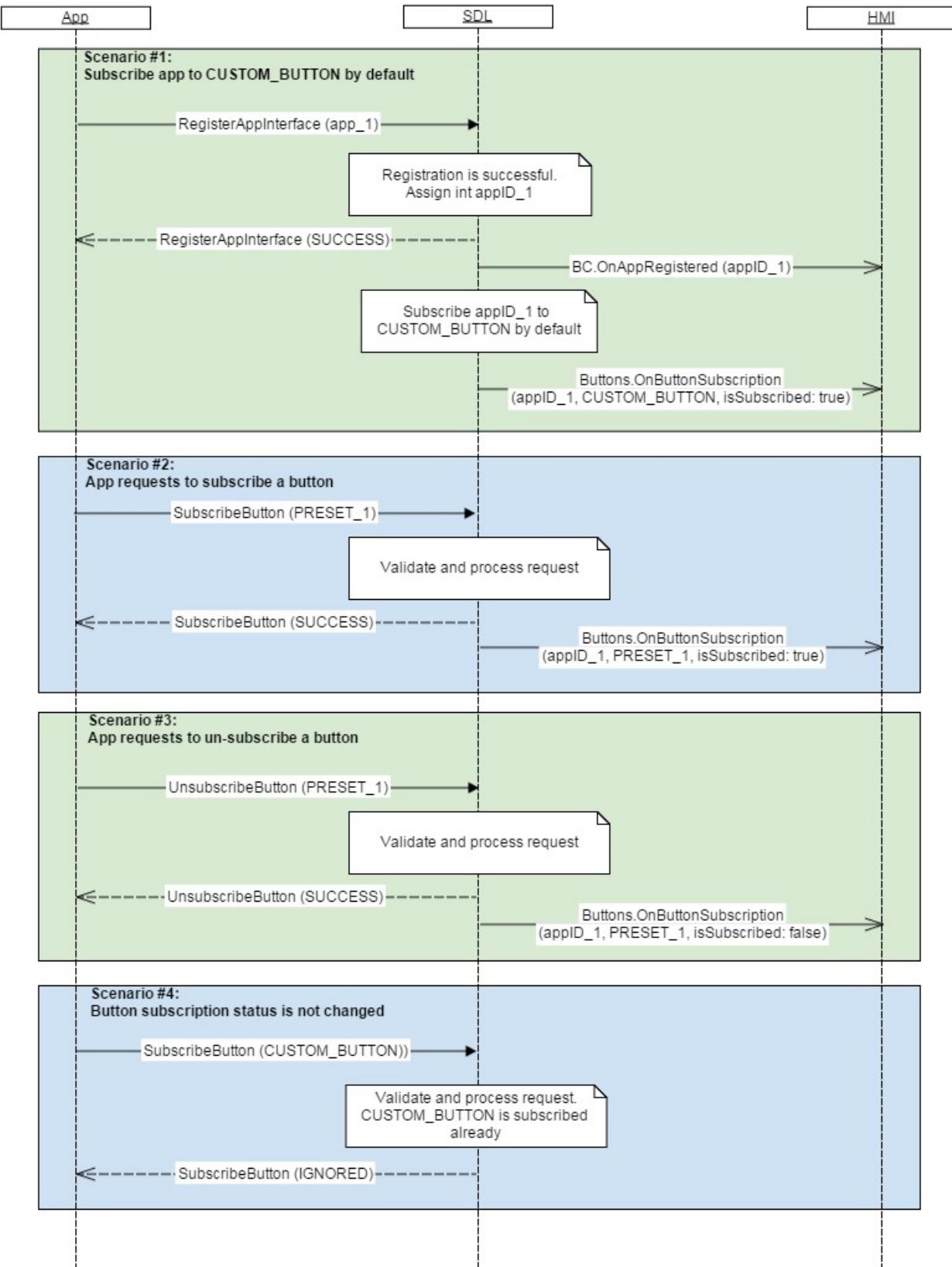
8.4.1.2 ButtonName

Element name	Value	Short Description
OK	0	Represents the button usually labeled "OK". A typical use of this button is for the User to press it to make a selection.
SEEKLEFT	1	Represents the seek-left button. A typical use of this button is for the user to scroll to the left through menu choices, one menu item per press.
SEEKRIGHT	2	Represents the seek-right button. A typical use of this button is for the user to scroll to the right through menu choices one menu item per press.

Element name	Value	Short Description
TUNEUP	3	Represents a turn of the tuner knob in the clockwise direction one tick.
TUNEDOWN	4	Represents a turn of the tuner knob in the counter-clockwise direction one tick.
PRESET_0	5	Represents the preset 0 button.
PRESET_1	6	Represents the preset 1 button.
PRESET_2	7	Represents the preset 2 button.
PRESET_3	8	Represents the preset 3 button.
PRESET_4	9	Represents the preset 4 button.
PRESET_5	10	Represents the preset 5 button.
PRESET_6	11	Represents the preset 6 button.
PRESET_7	12	Represents the preset 7 button.
PRESET_8	13	Represents the preset 8 button.
PRESET_9	14	Represents the preset 9 button.
CUSTOM_BUTTON	15	Represents any of onscreen buttons requested by SDL.
SEARCH	16	Represents the 'Search' button on the touchscreen keyboard.
PLAY_PAUSE	17	Represents Play/Pause button functionality (e.g. Play and Pause audio source)

8.4.2 Sequence Diagrams

8.4.2.1 OnButtonSubscription



8.4.3 JSON Messages Examples

{

```

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

```

9 VR Component Description

9.1 IsReady

9.1.1 Description

Type:	Function
Sender:	SDL
Purpose:	Know if UI module is ready.

The request comes after HMI's readiness is confirmed via [OnReady](#) notification. SDL requires the information about whether the voice recognition module is physically present on HU and ready to communicate with SDL.

Note:

If VR module is responded to be unavailable or IsReady request is not responded at all, SDL will not further send any requests related to it.

9.1.2 Request

7.1.2.1 Behavior

HMI must:

- 1) Check whether VR module is present and ready
- 2) Respond correspondingly to results of this check.

9.1.3 Response

Note:

There is a difference in message type for WebSocket and D-Bus connection (see the table below).

Result	Description	Message type		Message Params	Notes
		WebSocket	D-Bus		
Success	SUCCESS: HMI provides the information about VR module availability.	JSON response	Method return	available, code: 0	
Failure	INVALID_DATA: The data sent is invalid (invalid JSON syntax)	JSON <u>error message</u>	Method return	code: 11	Applicable for this RPC result codes.

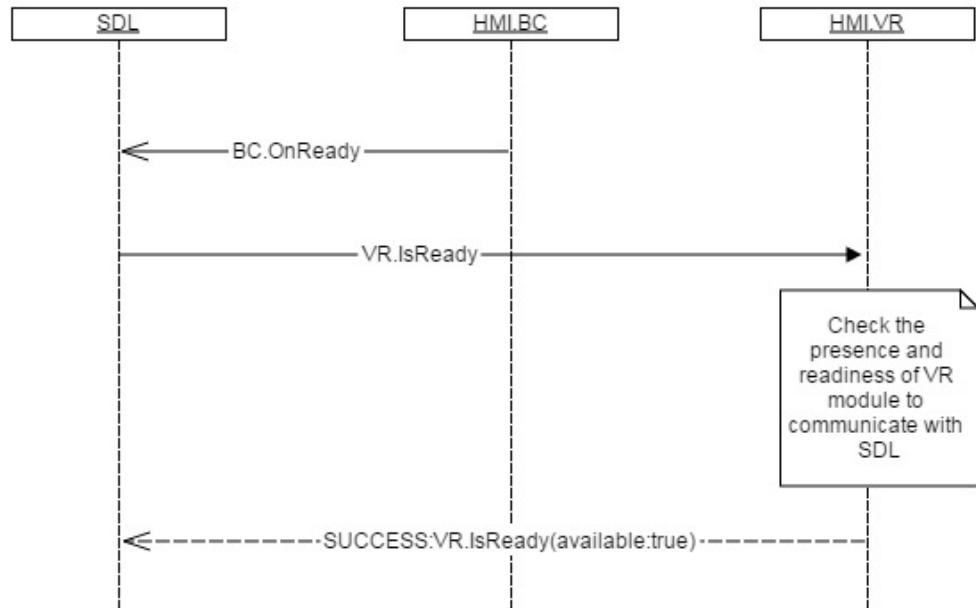
	<p>DATA_NOT_AVAILABLE: The information about VR module availability cannot be provided.</p>		Code: 9	Please see Result Enumeration for all SDL-supported codes.
	<p>GENERIC_ERROR: The unknown issue occurred or other codes are not applicable.</p>		code: 22	

9.1.3.1 Parameters

Param Name	Type	Mandatory	Description
available	Boolean	true	Must be - 'true' if VR module is present and ready to communicate with SDL - 'false' if not.

9.1.4 Sequence Diagrams

9.1.4.1 VR.IsReady



9.1.5 JSON Messages Examples

9.1.5.1 Request

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

9.1.5.2 Response

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

9.1.5.3 Error message

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

9.2 GetCapabilities

9.2.1 Description

Type:	Function
Sender:	SDL
Purpose:	Get capabilities of VR module

SDL can store the HMI capabilities in `smartDeviceLink.ini` file saved in SDL directory (see `HMICapabilities` parameter of ini file). Nevertheless, on its startup after getting `OnReady` notification and response to `VR.IsReady` from HMI, SDL sends this RPC for obtaining the capabilities of VR.

SDL Note: In case SDL didn't get VR capabilities from HMI, it will use the default ones from the file `hmi_capabilities.json`. The path to the file is defined in `smartDeviceLink.ini` as `HMICapabilities` parameter. See also [15.3.2.2 VR section of hmi_capabilities.json](#)

9.2.2 Request

9.2.2.1 Behavior

HMI must:

- 1) Check the VR capabilities.
- 2) Respond correspondingly to results of this check.

Note:

Currently only *TEXT* is SDL-supported capability of VR.

9.2.3 Response

Note:

There is a difference in message type for WebSocket and D-Bus connection (see the table below).

Result	Description	Message type		Message Params	Notes
		WebSocket	D-Bus		
Success	SUCCESS: HMI provides the VR capabilities.	JSON response	Method return	vrCapabilities, code: 0	
Failure	INVALID_DATA: The data sent is invalid (invalid JSON syntax)	JSON error message	Method return	code: 11	Applicable for this RPC result codes. Please see Result Enumeration for all SDL-supported codes.
	DATA_NOT_AVAILABLE: The information about VR capabilities cannot be provided.			Code: 9	
	GENERIC_ERROR: The unknown issue occurred or other codes are not applicable.			code: 22	

9.2.3.1 Parameters

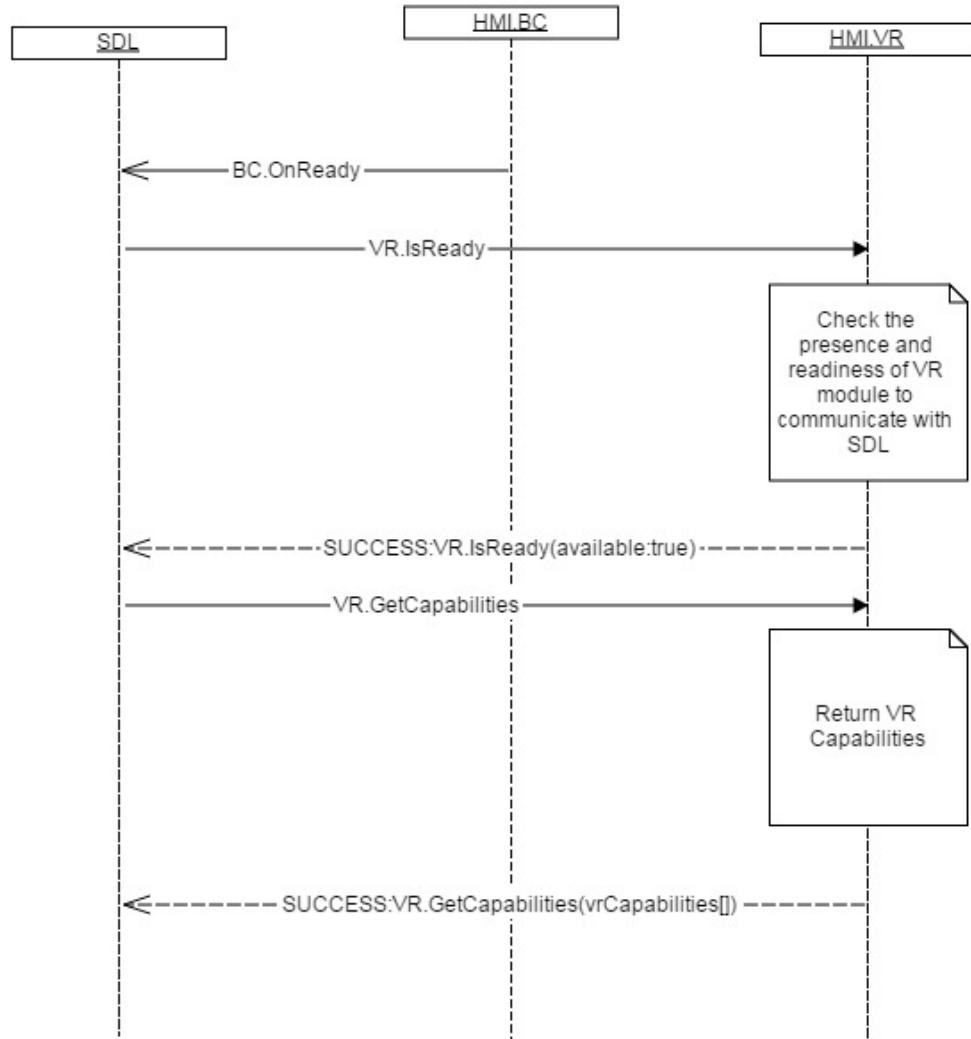
Param name	Type	Mandatory	Additional	Description
vrCapabilities	Common.VrCapabilities	false	Array = true minsize = 1 maxsize = 100	Types of input recognized by VR module.

9.2.3.2 VrCapabilities

Element name	Short Description
TEXT	Text is supported

9.2.4 Sequence Diagrams

9.2.4.1 VR.GetCapabilities



9.2.5 JSON Messages Examples

9.2.5.1 Request

```
{
  "id" : 9,
  "jsonrpc" : "2.0",
  "method" : "VR.GetCapabilities"
}
```

9.2.5.2 Response

```
{
  "id" : 9,
  "jsonrpc" : "2.0",
  "result" :
  {
    "vrCapabilities" : [TEXT],
    "code" : 0,
    "method" : "VR.GetCapabilities"
  }
}
```

9.2.5.3 Error message

```
{  
    "id" : 9,  
    "jsonrpc" : "2.0",  
    "error" :  
    {  
        "code" : 11,  
        "message" : "The data sent is  
invalid",  
        "data" :  
        {  
            "method" :  
"VR.GetCapabilities"  
        }  
    }  
}
```

9.3 GetSupportedLanguages

9.3.1 Description

Type:	Function
Sender:	SDL
Purpose:	Get the list of languages supported on VR

Once VR module is confirmed to be ready (via response to [IsReady](#) RPC) SDL starts discovering its capabilities via [GetCapabilities](#) and [GetSupportedLanguages](#) RPCs.

Response to [VR.GetSupportedLanguages](#) is assumed to bring the information about what languages are supported for voice recognition by VR module. Having obtained this information SDL will monitor the language parameter within RPCs from mobile application(s) and reject the requests containing language not supported by HMI.

Note:

The list of languages recognized by SDL is provided in the section 9.3.3.2 Language Enumeration.

9.3.2 Request

7.1.2.1 Behavior

HMI must:

- 1) Check the VR supported languages
- 2) Respond correspondingly to results of this check.

9.3.3 Response

Note:

There is a difference in message type for WebSocket and D-Bus connection (see the table below).

Result	Description	Message type	Message Params	Notes
--------	-------------	--------------	----------------	-------

		WebSocket	D-Bus		
Success	SUCCESS: HMI provides the VR supported languages.	JSON response	Method return	language, code: 0	
Failure	INVALID_DATA: The data sent is invalid (invalid JSON syntax)	JSON error message	Method return	code: 11	Applicable for this RPC result codes. Please see Result Enumeration for all SDL-supported codes.
	DATA_NOT_AVAILABLE: The information about VR supported languages cannot be provided.			Code: 9	
	GENERIC_ERROR: The unknown issue occurred or other codes are not applicable.			code: 22	

9.3.3.1 Parameters

Param name	Type	Mandatory	Additional	Description
languages	Common.Language	true	Array = true minsize = 1 maxsize = 100	List of languages supported in VR. See Language.

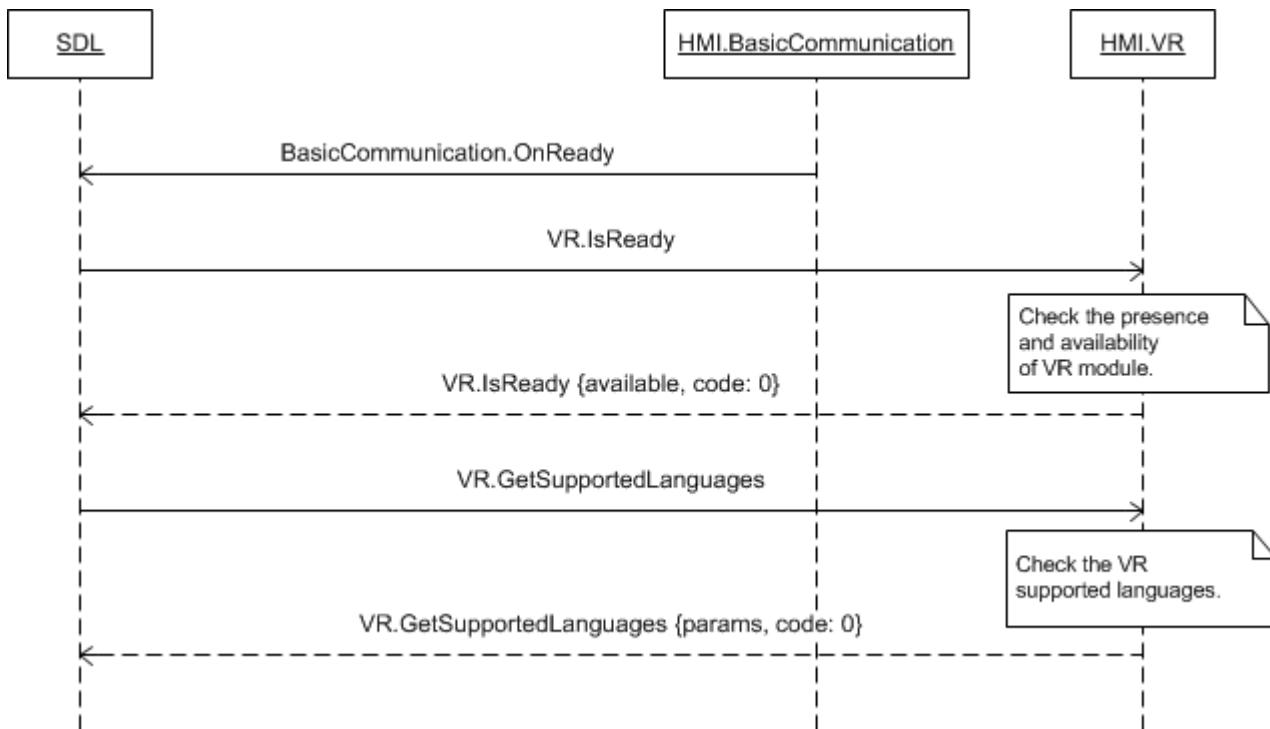
9.3.3.2 Language Enumeration

Element name	Short Description
AR-SA	Arabic – Saudi Arabia
CS-CZ	Czech – Czech Republic
DA-DK	Danish – Denmark
DE-DE	German – Germany
EN-AU	English – Australia
EN-GB	English – GB
EN-US	English – US
ES-ES	Spanish – Spain
ES-MX	Spanish – Mexico
FR-CA	French – Canada
FR-FR	French – France
IT-IT	Italian – Italy
JA-JP	Japanese – Japan
KO-KR	Korean – South Korea
NL-NL	Dutch (Standard) – Netherlands
NO-NO	Norwegian - Norway
PL-PL	Polish – Poland
PT-PT	Portuguese – Portugal
PT-BR	Portuguese – Brazil
RU-RU	Russian - Russia
SV-SE	Swedish – Sweden
TR-TR	Turkish – Turkey

Element name	Short Description
ZH-CN	Mandarin – China
ZH-TW	Mandarin – Taiwan
NL-BE	Dutch Belgium (Flemish)
EL-GR	Greek
HU-HU	Hungarian
FI-FI	Finnish
SK-SK	Slovak

9.3.4 Sequence Diagrams

9.3.4.1 VR.GetSupportedLanguages



9.3.5 JSON Messages Examples

9.3.5.1 Request

```
{
  "id" : 18,
  "jsonrpc" : "2.0",
  "method" : "VR.GetSupportedLanguages"
}
```

9.3.5.2 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"
}
}

```

9.3.5.3 Error message

```

{
    "id" : 18,
    "jsonrpc" : "2.0",
    "error" :
    {
        "code" : 11,
        "message" : "The data sent is
invalid",
        "data" :
        {
            "method" :
"VR.GetSupportedLanguages"
        }
    }
}

```

9.4 AddCommand

9.4.1 Description

Type:	Function
Sender:	SDL
Purpose:	Add a command for voice recognition.

SDL requests to add a command for voice recognition module.

9.4.2 Request

9.4.2.1 Behavior

HMI must:

- 1) Store the provided values of cmdID and corresponding vrCommands.
- 2) Keep these values stored until VR.DeleteCommand or BasicCommunication.OnAppUnregistered RPC comes from SDL.
- 3) Add the vrCommands to voice recognition system and make them accessible in the following layers:
 - If appId parameter is present in request: HMI must use the vrCommands in the layer of the appId application (i.e. when it is active on HMI).
 - If appId parameter is not present in request: HMI must use the vrCommands in all layers.
- 4) Manage grammarID as an identifier for compiled VR grammar for a command/ set of commands (a command and its synonyms or a choiceSet). So in the VR engine's database, there may be multiple cmdID entries under each unique grammarID.

Important note:

HMI must return grammarID and cmdId via VR.OnCommand when the corresponding synonym is recognized.

In case the app sends AddCommand with the both UI and VR portions and adding one of the portions ended with erroneous response or no response at all, SDL must send DeleteCommand for a successfully added portion(VR or UI) in the following cases (see diagrams for more details [9.4.4.2-9.4.2.5](#)) :

1. In case SDL sends both UI.AddCommand and VR.AddCommand with one and the same cmdID to HMI AND **UI.AddCommand gets successful response** from HMI in return AND VR.AddCommand gets any *erroneous response*/no response from HMI - SDL must send **UI.DeleteCommand** for the successfully added cmdID to HMI.
2. In case SDL sends both UI.AddCommand and VR.AddCommand with one and the same cmdID to HMI AND **VR.AddCommand gets successful response** from HMI in return AND UI.AddCommand gets *erroneous response except of WARNINGS and UNSUPPORTED_RESOURCE*/no response from HMI - SDL must send **VR.DeleteCommand** for the successfully added cmdID to HMI

VR.AddCommand in case of CreateInteractionChoiceSet:

1. VR.AddCommands for every ChoiceSet is sent by SDL after the app requests CreateInteractionChoiceSet. The subsequent VR.AddCommands are issued for every choice of the ChoiceSet and once a response (positive or negative) is returned by the HMI / Voice Engine, then the app is informed by SDL if the Choice Set is ready (i.e. grammar is built on the platform and can be referenced in an interaction).
2. When SDL sends the set of VR.AddCommands (as a part of mobile API CreateInteractionChoiceSet) and HMI returns failure to any VR.AddCommand from this set - SDL will send VR.DeleteCommands to all previously successfully added commands of this set. Set of such commands has the same grammarID.

[9.4.2.2 Parameters](#)

Param name	Type	Mandatory	Additional	Description
cmdId	Integer	true	minvalue = 0	Unique ID that identifies the command. Must be returned in the OnCommand notification to identify the command selected by the User.

Param name	Type	Mandatory	Additional	Description
vrCommands	String	true	minsize = 1 maxsize = 100 maxlength = 99 array = true	An array of strings to be used as VR synonyms for this command. Defines one or more VR phrases the recognition of any of which must trigger the <code>OnCommand</code> notification with the above <code>cmdID</code> .
type	Common.VRCCommandType	true	-	Type of added command. See <code>VRCCommandType</code> .
grammarID	Integer	true	minvalue = 0 maxvalue = 2000000000	ID of the specific grammar, whether top-level or choice set. <code>grammarID</code> is a unique value within each application For all choices within a choice set <code>grammarID</code> is the same
appID	Integer	false	-	ID of the application related to this RPC.

9.4.2.3 `VRCCommandType`

This type defines whether the command relates to common (top-level) list of the application commands or it's a part of the ChoiceSet which will be available to the user during PerformInteraction

Element name	Vlaue	Short Description
Choice	0	The VR command must be used in VR.PerformInteraction that arrived with the corresponding grammarID and must NOT be accessible until then.
Command	1	The VR command must be accessible for the User upon VR activation while the application of the corresponding appID is active on HMI.

9.4.3 Response

Note:

There is a difference in message type for WebSocket and D-Bus connection (see the table below).

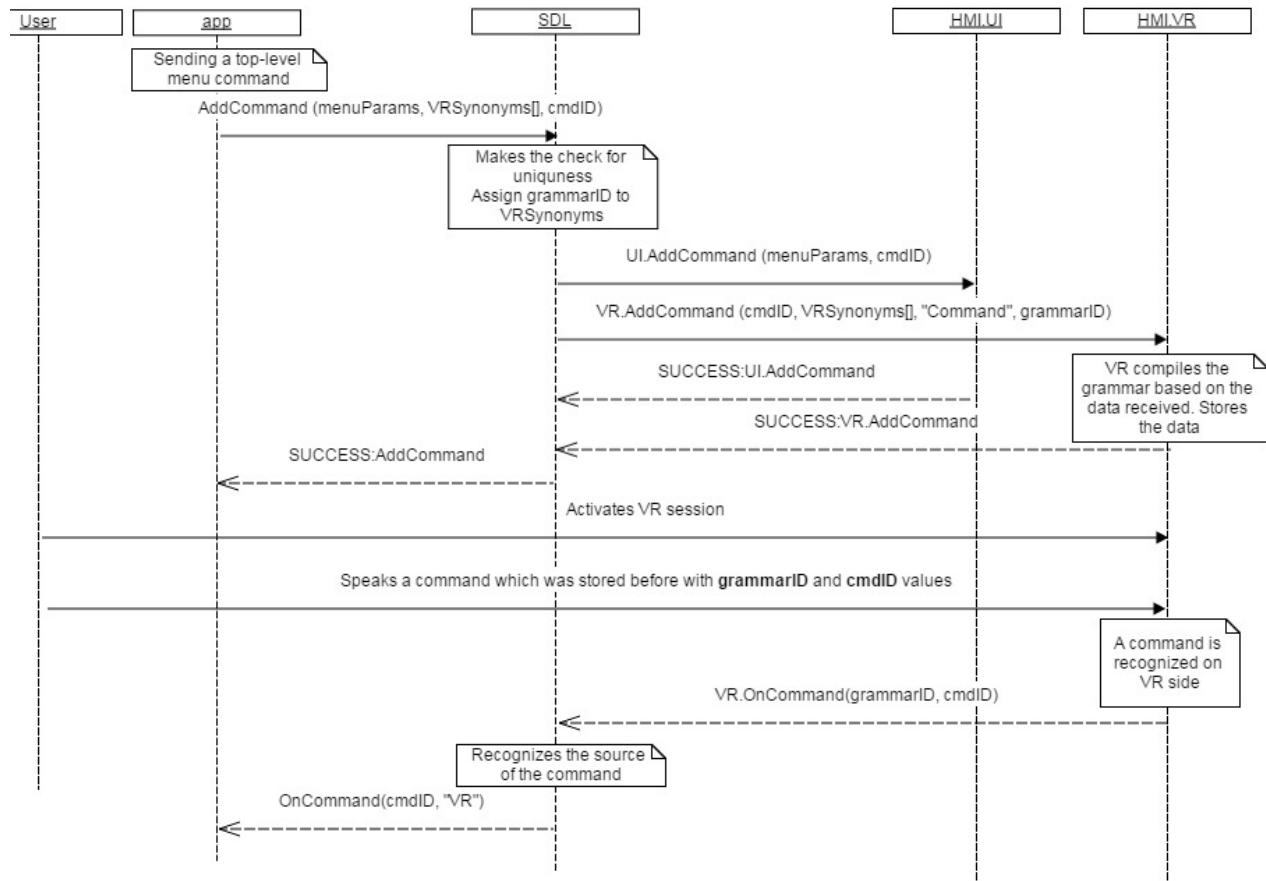
Result	Description	Message type		Message Params	Notes
		WebSocket	D-Bus		
Success	SUCCESS: The command with requested parameters has been added for voice recognition for the named application.	JSON response	Method return	code: 0	
Failure	INVALID_DATA: The data sent is invalid (invalid JSON syntax, out of bound values)	JSON error message	Method return	code: 11	INVALID_ID check is performed on SDL side. In case HMI performs the

	INVALID_ID appId, cmdID, grammarID is invalid (e.g. doesn't exist or already exist) GENERIC_ERROR 1) The unknown issue occurred or other codes are not applicable.		code: 13 code: 22	check on it's side too, the appropriate code must be return.
--	---	--	--------------------------	--

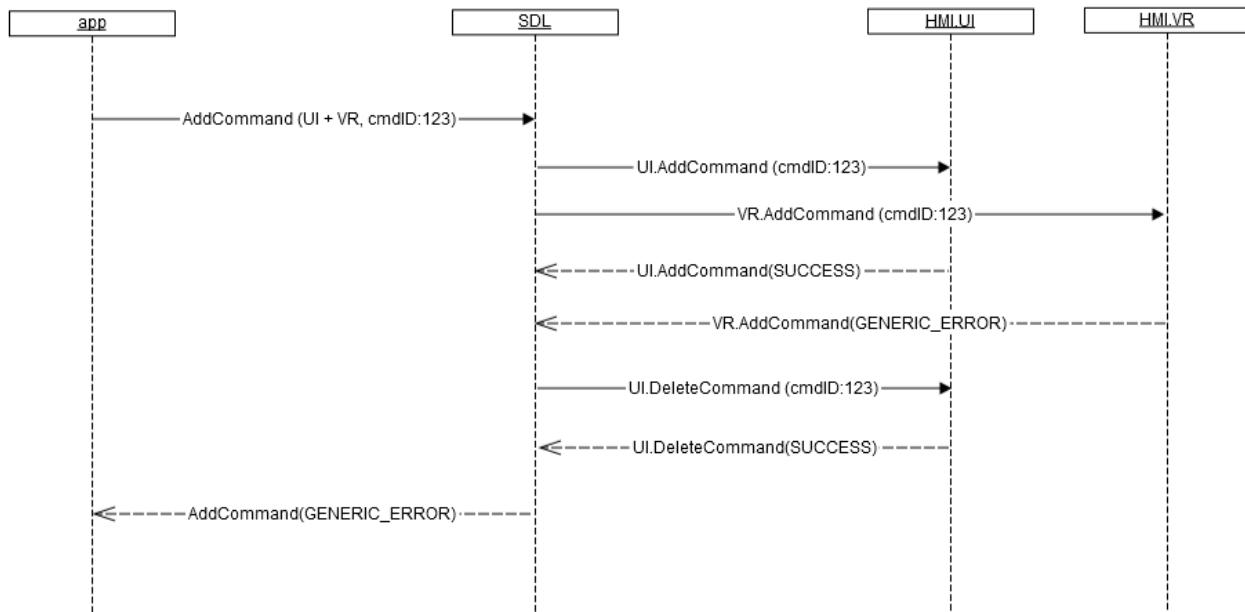
SDL note: In case HMI does not respond SDL's request during SDL default timeout (10 sec), SDL will return GENERIC_ERROR result code to the corresponding mobile app's request.

9.4.4 Sequence Diagrams

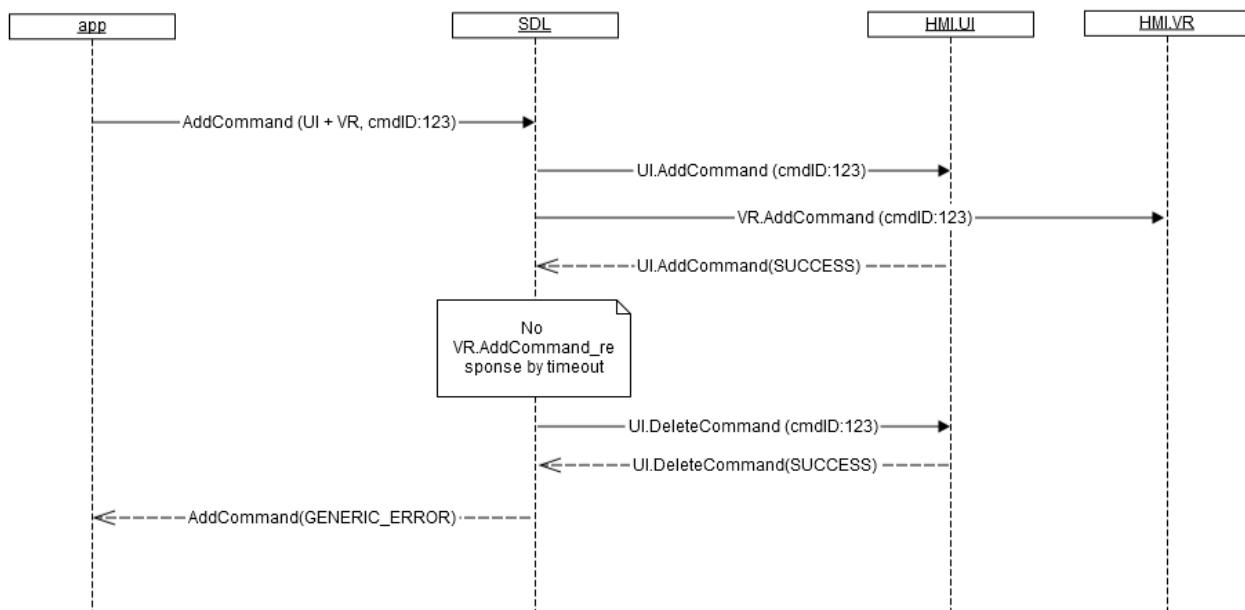
9.4.4.1 VR.AddCommand



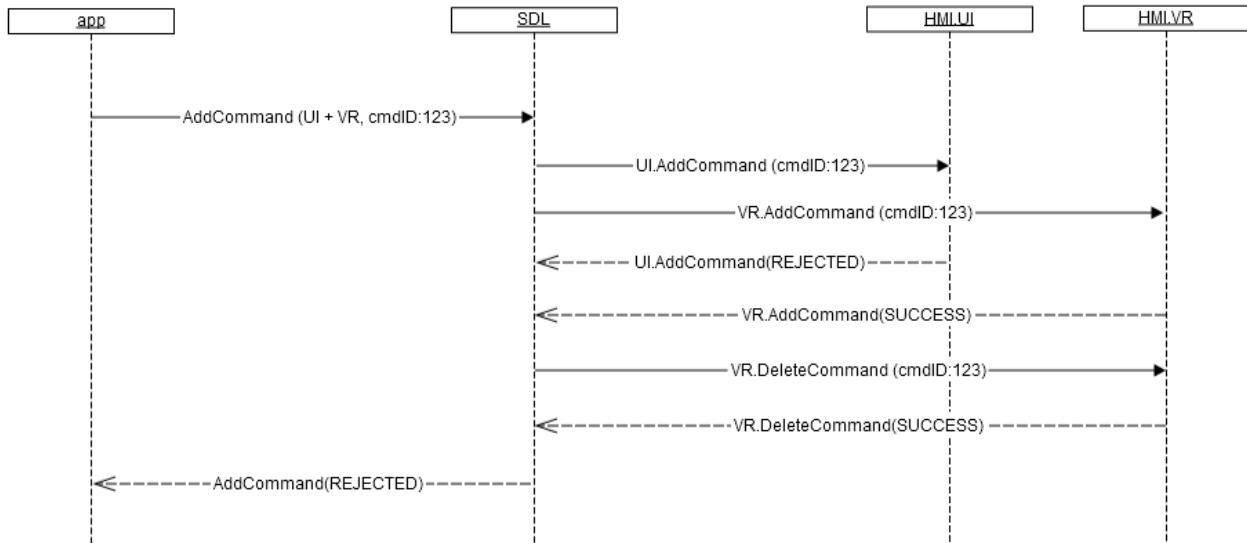
**9.4.4.2 UI.AddCommand returns SUCCESS,
VR portion failed**



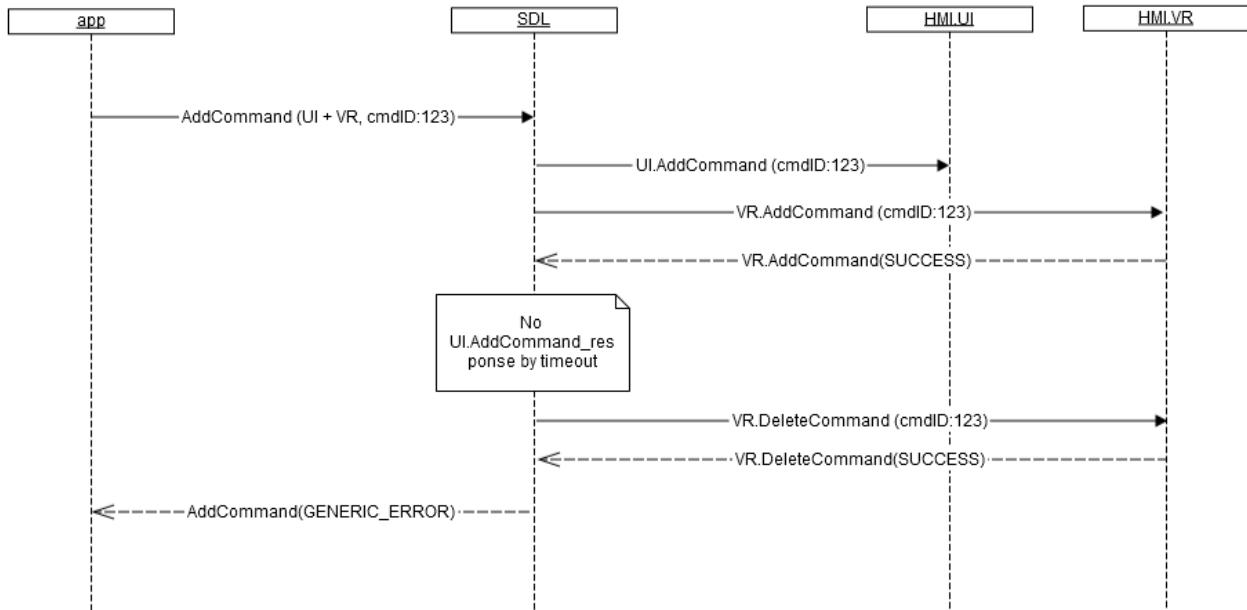
**9.4.4.3 UI.AddCommand returns SUCCESS,
VR portion no response**



9.4.4.4 UI.AddCommand fails, VR portion SUCCESS



9.4.4.5 UI.AddCommand no response, VR portion SUCCESS



9.4.5 JSON Messages Examples

9.4.5.1 Request

```
{
  "id" : 119,
  "jsonrpc" : "2.0",
  "method" : "VR.AddCommand",
```

```

"params" :
{
    "cmdID" : 4365,
    "vrCommands" :
    [
        "Leave",
        "Exit",
        "Quit"
    ],
    "grammarID":123,
    "appID" : 64467
}
}

```

9.4.5.2 Response

```

{
    "id" : 119,
    "jsonrpc" : "2.0",
    "result" :
    {
        "code" : 0,
        "method" : "VR.AddCommand"
    }
}

```

9.4.5.3 Error message

```

{
    "id" : 119,
    "jsonrpc" : "2.0",
    "error" :
    {
        "code" : 13,
        "message" : "Provided appID is not
valid",
        "data" :
        {
            "method" : "VR.AddCommand"
        }
    }
}

```

9.5 DeleteCommand

9.5.1 Description

Type:	Function
Sender:	SDL
Purpose:	Delete a command from VR.

SDL requests to delete the command from in-application menu or sub menu of the named application previously added via [VR.AddCommand](#).

The request may arrive in both cases of activated and deactivated application on HMI (depends on Policy Table permissions applicable to mobile application request, by

default allowed to operate on all HMI levels except of NONE).

9.5.2 Request

9.5.2.1 Behavior

HMI must:

- 1) Update the named application stored data correspondingly to the RPC arrived.
- 2) Delete the command identified with cmdID and grammarID.
- 3) Display updates if the named application is active and VR Help list is open on UI (in case received updates by SetGlobalProperties)
- 4) Provide the response corresponding to the result of RPC execution.

Note:

The applicable to this RPC result codes are provided in section 9.5.3 Response.

Note:

- The value of cmdID and grammarID were previously sent via VR.AddCommand.
- The value of appID is previously sent via UpdateAppList or OnAppRegistered.

Note about CreateInteractionChoiceSet and

VR.AddCommand: When SDL sends a set of VR.AddCommands (as a part of mobile API CreateInteractionChoiceSet) **and** HMI returns failure to any VR.AddCommand from this set. SDL will send VR.DeleteCommands to all previously successfully added commands of this set. Set of such commands has the same grammarID.

9.5.2.2 Parameters

Param Name	Type	Mandatory	Additional	Description
cmdId	Integer	true	-	ID that identifies the command to be deleted (sent by AddCommand).
type	Common.VRCommandType	true	-	Type of added command. See VR CommandType.
grammarID	Integer	true	minvalue = 0 maxvalue = 2000000000	ID of the specific grammar, whether top-level or choice set.
appID	Integer	true	-	ID of application that concerns this RPC.

9.5.3 Response

Note:

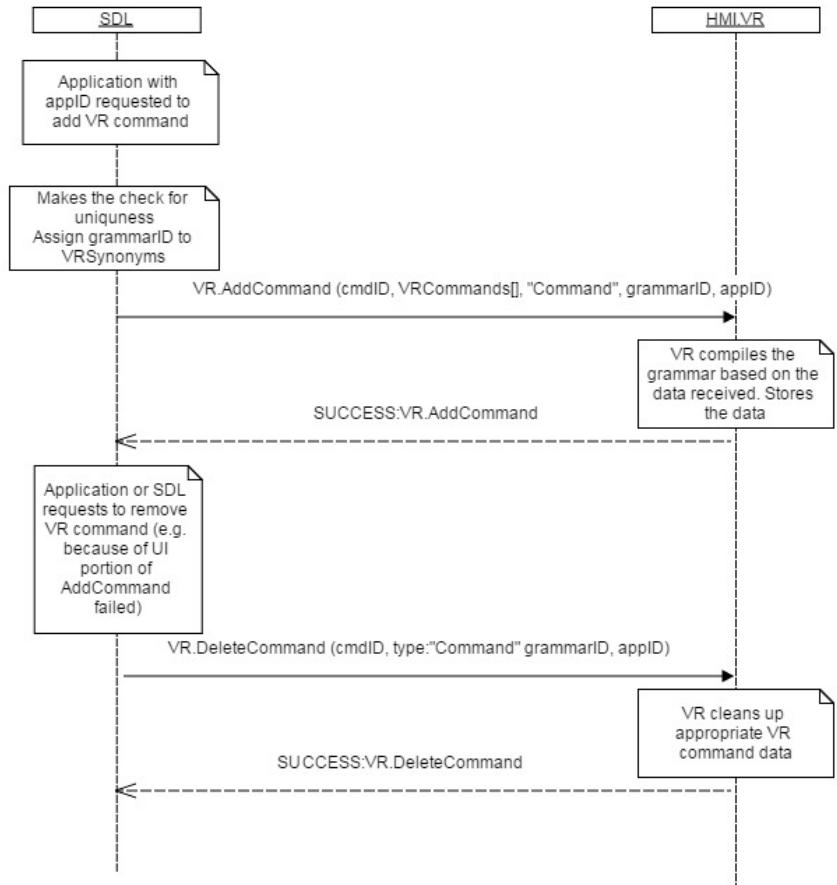
There is a difference in message type for WebSocket and D-Bus connection (see the table below).

Result	Description	Message type		Message Params	Notes
		WebSocket	D-Bus		
Success	SUCCESS: The named command has been deleted from VR for the named application.	JSON response	Method return	code: 0	
Failure	INVALID_DATA: The data sent is invalid (invalid JSON syntax, out of bounds parameters)	JSON error message	Method return	code: 11	Applicable for this RPC result codes. Please see Result Enumeration for all SDL-supported codes.
	INVALID_ID cmdId, grammarID or appId is not valid (e.g. doesn't exist)			code: 13	
	GENERIC_ERROR 1) The unknown issue occurred or other codes are not applicable.			code: 22	

SDL 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.

9.5.4 Sequence Diagrams

9.1.4.1 VR.DeleteCommand



9.5.5 JSON Messages Examples

9.5.5.1 Request

```
{
  "id" : 147,
  "jsonrpc" : "2.0",
  "method" : "VR.DeleteCommand",
  "params" :
  {
    "cmdID" : 4365,
    "type": "Command",
    "grammarID": 13,
    "appId" : 8764
  }
}
```

9.5.5.2 Response

```
{
  "id" : 147,
  "jsonrpc" : "2.0",
  "result" :
  {
    "code" : 0,
    "method" : "VR.DeleteCommand"
  }
}
```

9.5.5.3 Error message

```
{  
    "id" : 147,  
    "jsonrpc" : "2.0",  
    "error" :  
    {  
        "code" : 13,  
        "message" : "One of the provided IDs  
is not valid",  
        "data" :  
        {  
            "method" :  
"VR.DeleteCommand"  
        }  
    }  
}
```

9.6 ChangeRegistration

9.6.1 Description

Type:	Function
Sender:	SDL
Purpose:	Change the application language for voice recognition on HMI.

SDL requests to set VR language for the named application on HMI.

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

Note:

SDL will send the language value in the request expected to be supported by application on HMI. The value will be obtain via VR.GetCapabilities or properties file(defined in smartDeviceLink.ini as HMICapabilities parameter) .

9.6.2 Request

9.6.2.1 Behavior

HMI must:

- 1) Store the provided information associating it with application's appID
- 2) Change the setting of VR module for an appropriate application
- 3) Respond to the request.

9.6.2.2 Parameters

Param Name	Type	Mandatory	Additional	Description
------------	------	-----------	------------	-------------

Param Name	Type	Mandatory	Additional	Description
language	Common.Language	true		The language requested to be switched to. See Language.
vrSynonyms	String	false	array=true maxlength=40 minsize=1 maxsize=100	Request new VR synonyms registration Defines an additional voice recognition command. Must not interfere with any name of previously registered applications(SDL makes check).
appID	Integer	true		ID of the application that relates to this RPC.

9.6.2.2 Language

Element 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
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 Belgium (Flemish)
EL-GR	25	Greek
HU-HU	26	Hungarian

Element Name	Value	Description
FI-FI	27	Finnish
SK-SK	28	Slovak

9.6.3 Response

Note:

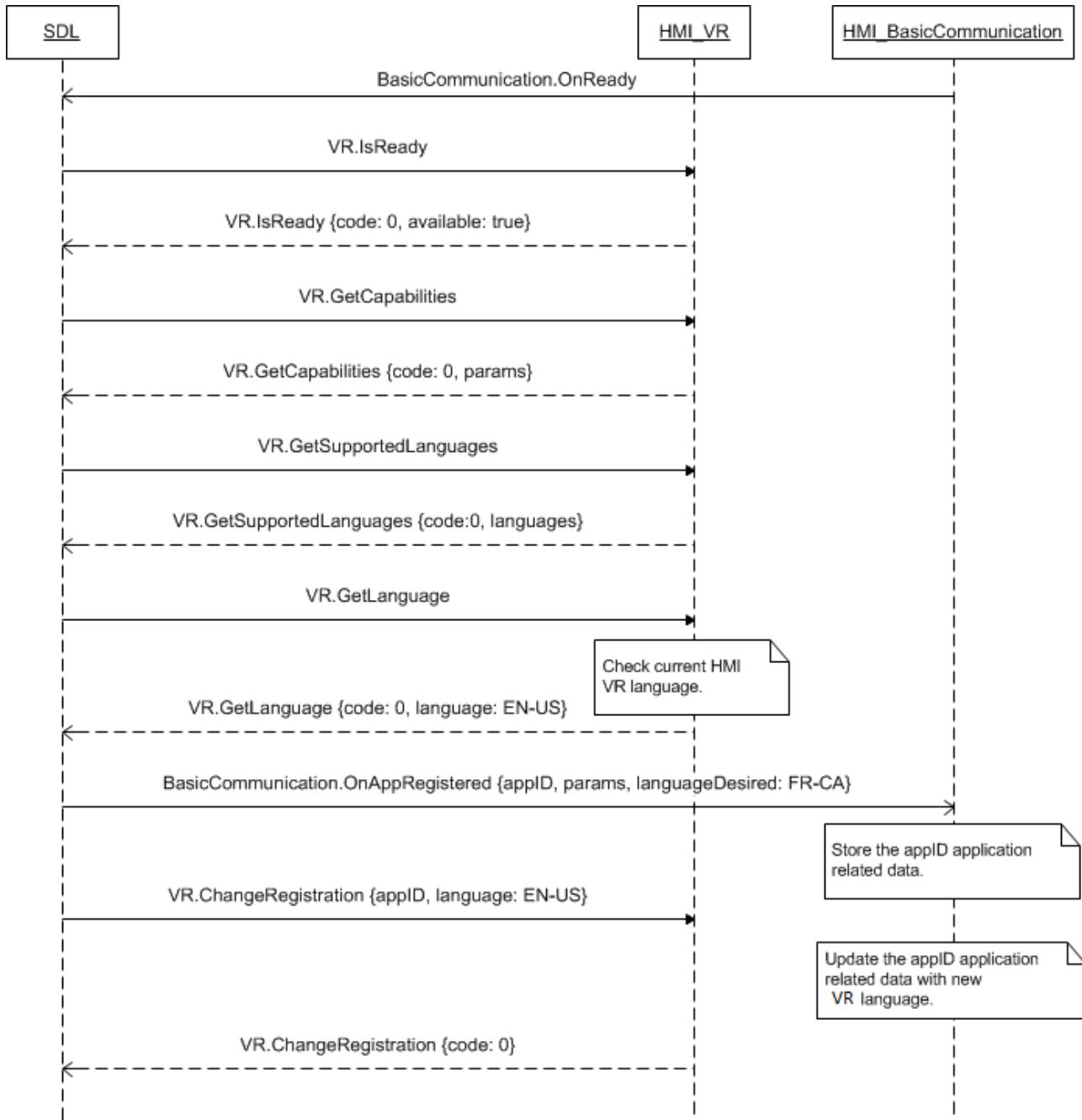
There is a difference in message type for WebSocket and D-Bus connection (see the table below).

Result	Description	Message type		Message Params	Notes
		WebSocket	D-Bus		
Success	SUCCESS: The requested language has been set for voice recognition for the named application.	JSON response	Method return	code: 0	
Failure	INVALID_DATA: The data sent is invalid (invalid JSON syntax, parameters are out of bounds or of wrong type)	JSON error message	Method return	code: 11	Applicable for this RPC result codes. Please see Result Enumeration for all SDL-supported codes.
	WRONG_LANGUAGE Language is not supported by VR			code: 16	
	INVALID_ID appId is invalid (e.g. doesn't exist)			code: 13	
	GENERIC_ERROR 1) The unknown issue occurred or other codes are not applicable. .			code: 22	

SDL 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

9.6.4 Sequence Diagrams

9.6.4.1 VR.ChangeRegistration



9.6.5 JSON Messages Examples

9.6.5.1 Request

```
{
    "id" : 206,
    "jsonrpc" : "2.0",
    "method" : "VR.ChangeRegistration",
    "params" :
    {
        "language" : "DE-DE",
        "appId" : 13264
    }
}
```

9.6.5.2 Response

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

9.6.5.3 Error message

```
{  
    "id" : 206,  
    "jsonrpc" : "2.0",  
    "error" :  
    {  
        "code" : 22,  
        "message" : "The unknown error  
occurred",  
        "data" :  
        {  
            "method" :  
            "VR.ChangeRegistration"  
        }  
    }  
}
```

9.7 GetLanguage

9.7.1 Description

Type:	Function
Sender:	SDL
Purpose:	Get the current language of VR

SDL inquires the current HMI VR language (general VR interface language, not relevant to application's VR languages).

This RPC is sent by SDL after VR readiness is confirmed by VR.IsReady. If later the User changes the HMI language of VR, HMI must inform SDL about this event via VR.OnLanguageChange notification.

9.7.2 Request

9.7.2.1 Behavior

HMI must:

- 1) Check the HMI VR language currently in effect.
- 2) Respond providing SDL with the results of this check.

9.7.3 Response

Note:

There is a difference in message type for WebSocket and D-Bus connection (see the table below).

Result	Description	Message type		Message Params	Notes
		WebSocket	D-Bus		
Success	SUCCESS: HMI provides the language currently active for VR.	JSON response	Method return	language, code: 0	
Failure	INVALID_DATA: The data sent is invalid (invalid JSON syntax, parameters are out of bounds or of wrong type)	JSON error message	Method return	code: 11	Applicable for this RPC result codes. Please see Result Enumeration for all SDL-supported codes.
	GENERIC_ERROR: The unknown issue occurred or other codes are not applicable.			code: 22	

9.7.3.1 Parameters

Param Name	Type	Mandatory	Description
language	Common.Language	true	The language requested to be switched to. See Language.

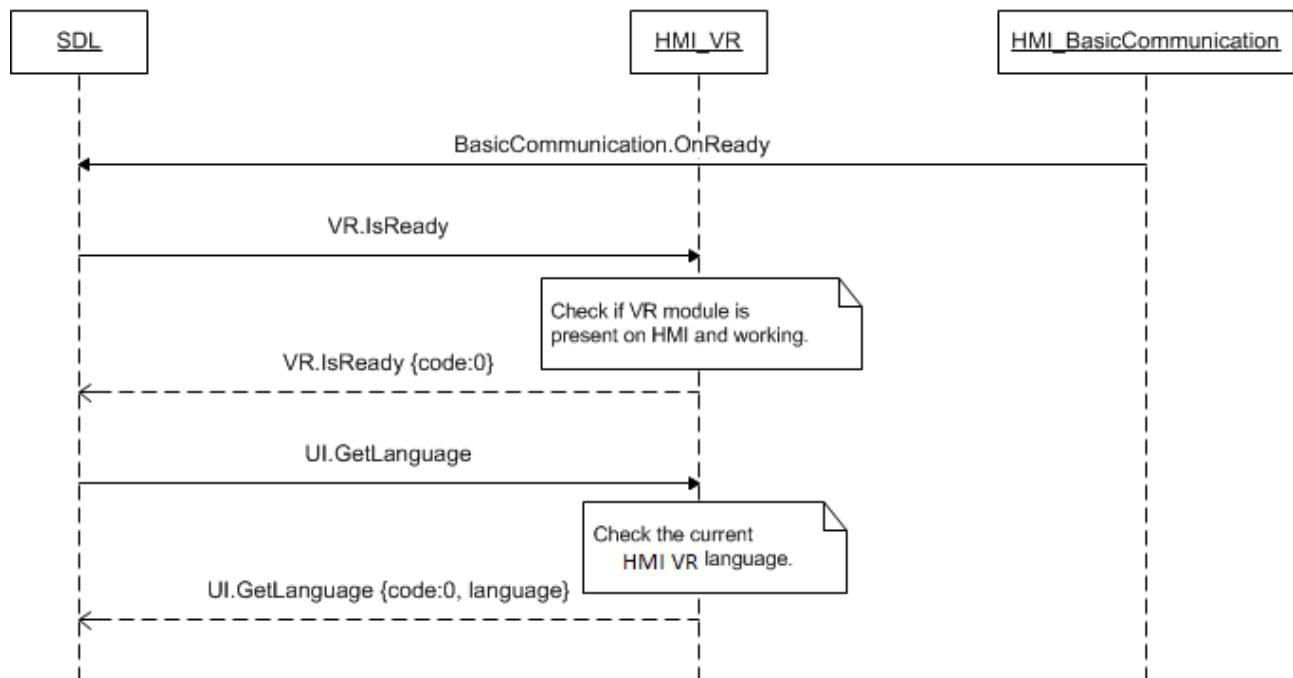
9.7.3.2 Language

Element 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

Element Name	Value	Description
ZH-TW	16	Mandarin – Taiwan
JA-JP	17	Japanese – Japan
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 Belgium (Flemish)
EL-GR	25	Greek
HU-HU	26	Hungarian
FI-FI	27	Finnish
SK-SK	28	Slovak

9.7.4 Sequence Diagrams

9.7.4.1 VR.GetLanguage



9.7.5 JSON Messages Examples

9.7.5.1 Request

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

9.7.5.2 Response

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

9.7.5.3 Error message

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

9.8 Started

9.8.1 Description

Type:	Notification
Sender:	HMI
Purpose:	Inform about start of VR session

Once VR is activated(ready to recognize User's speech), HMI should attenuate the audio or make it not audible (depending on its capabilities). SDL needs to be notified about the event in order to provide a mobile application with the accurate information about audio streaming state on HMI.

Note: SDL ignores all invalid notifications which come from HMI (invalid JSON, invalid data types/bounds etc)

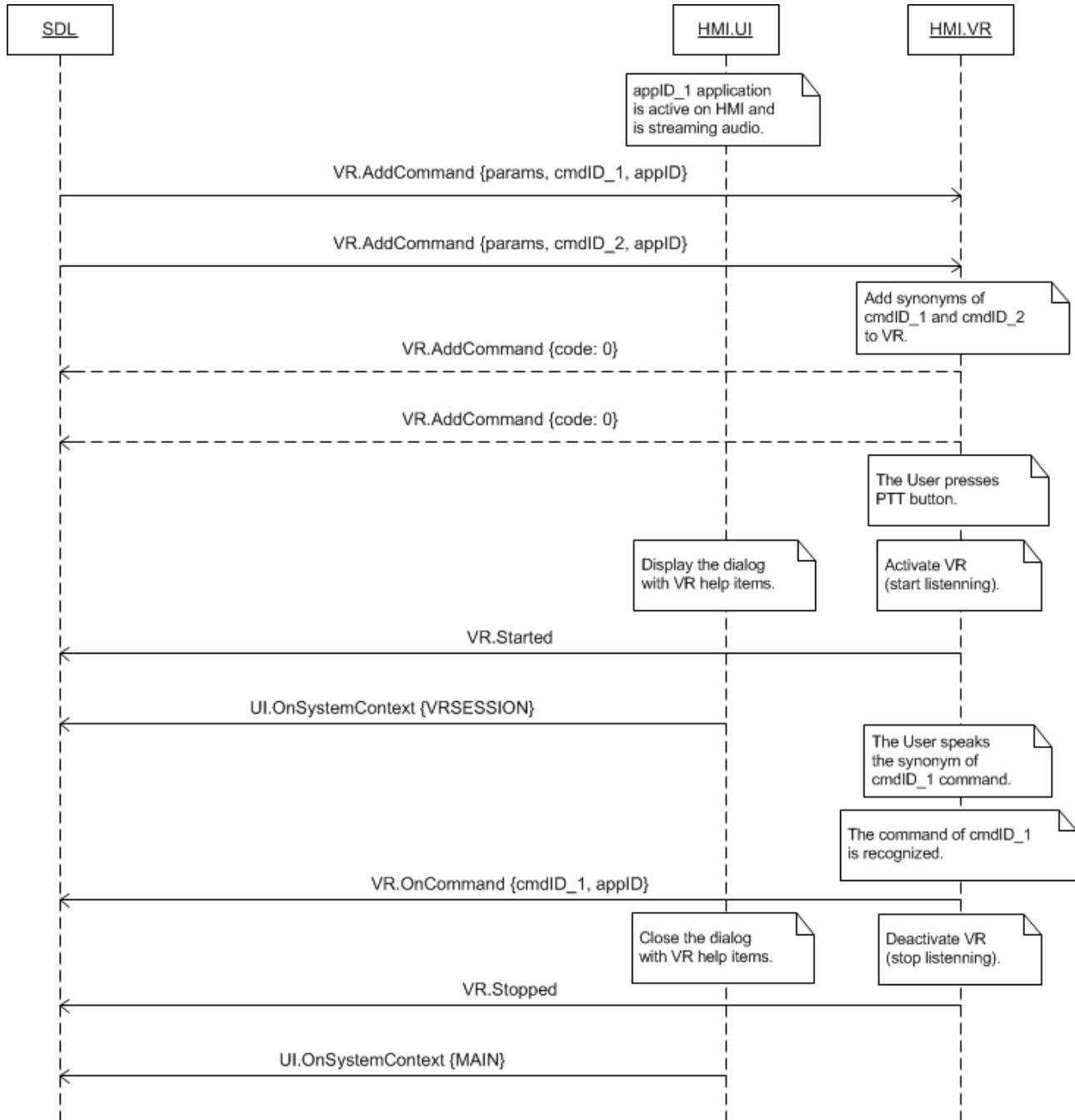
HMI must:

Send VR.Started notification when VR system is activated by:

- PTT button press
- VR.PerformInteraction RPC from SDL

9.8.2 Sequence Diagrams

9.8.2.1 VR.Started on PTT button press



9.8.3 JSON Messages Examples

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

9.9 Stopped

9.9.1 Description

Type:	Notification
Sender:	HMI
Purpose:	Inform about stop of VR session

After VR interaction with the User has completed, HMI should return the audio source to be audible for the User. SDL requires to be notified about the event so that to inform the mobile application about the actual audio streaming state on HMI.

Note: SDL ignores all invalid notifications which come from HMI (invalid JSON, invalid data types/bounds etc)

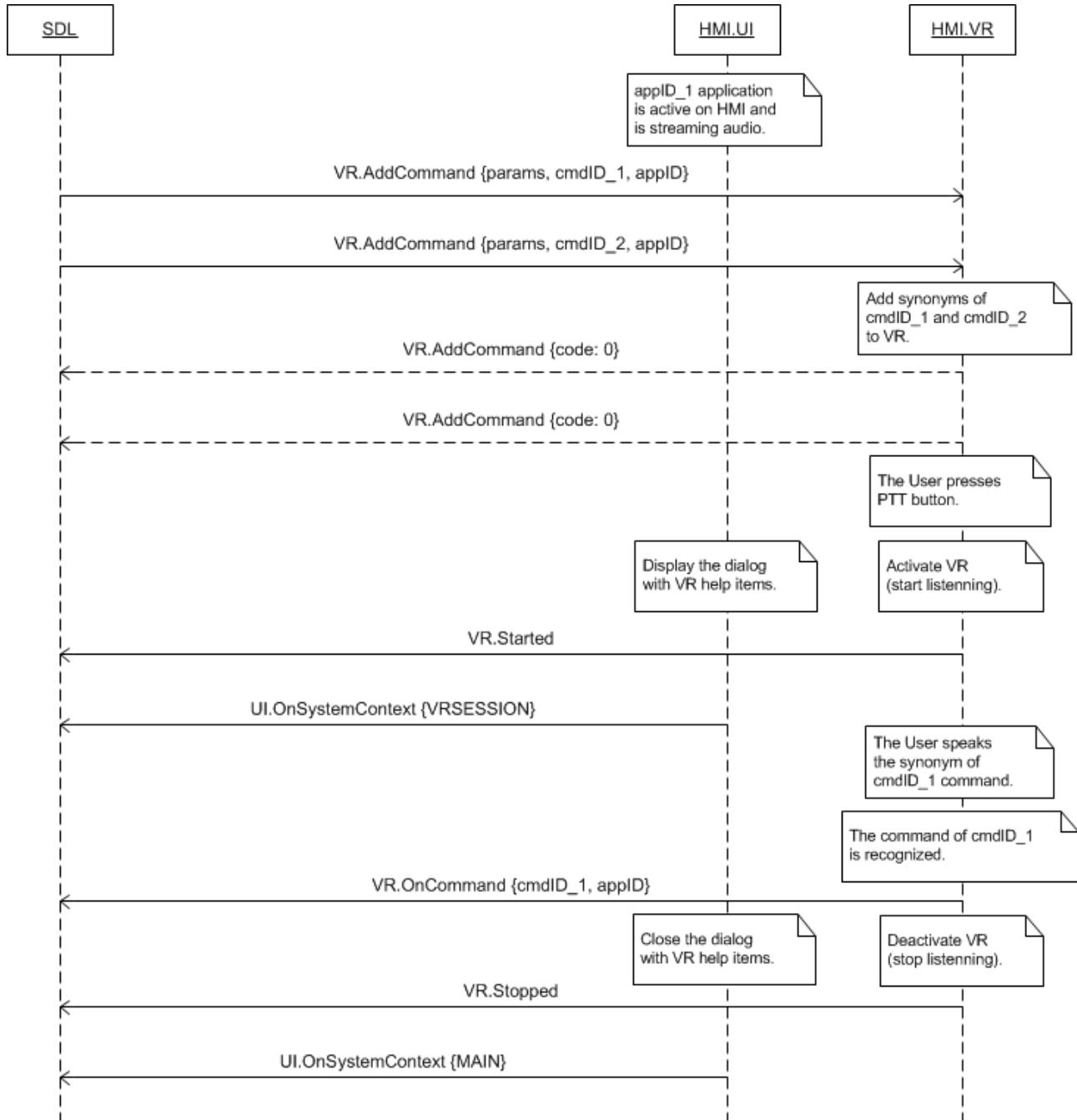
HMI must:

Send VR.Stopped notification when voice recognition is ended by:

- Successfully recognized command
- User cancelling the interaction
- Event of a higher priority

9.9.2 Sequence Diagrams

9.8.2.1 VR.Stopped on VR session ending



9.9.3 JSON Messages Examples

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

9.10 OnCommand

9.10.1 Description

Type:	Notification
Sender:	HMI
Purpose:	Inform the command is recognized by VR

OnCommand is used for informing about the user's choice of the application added command.

Note: SDL ignores all invalid notifications which come from HMI (invalid JSON, invalid data types/bounds etc)

HMI must:

- send a notification to SDL with all mandatory data as a result of recognition a VR command the user spoke

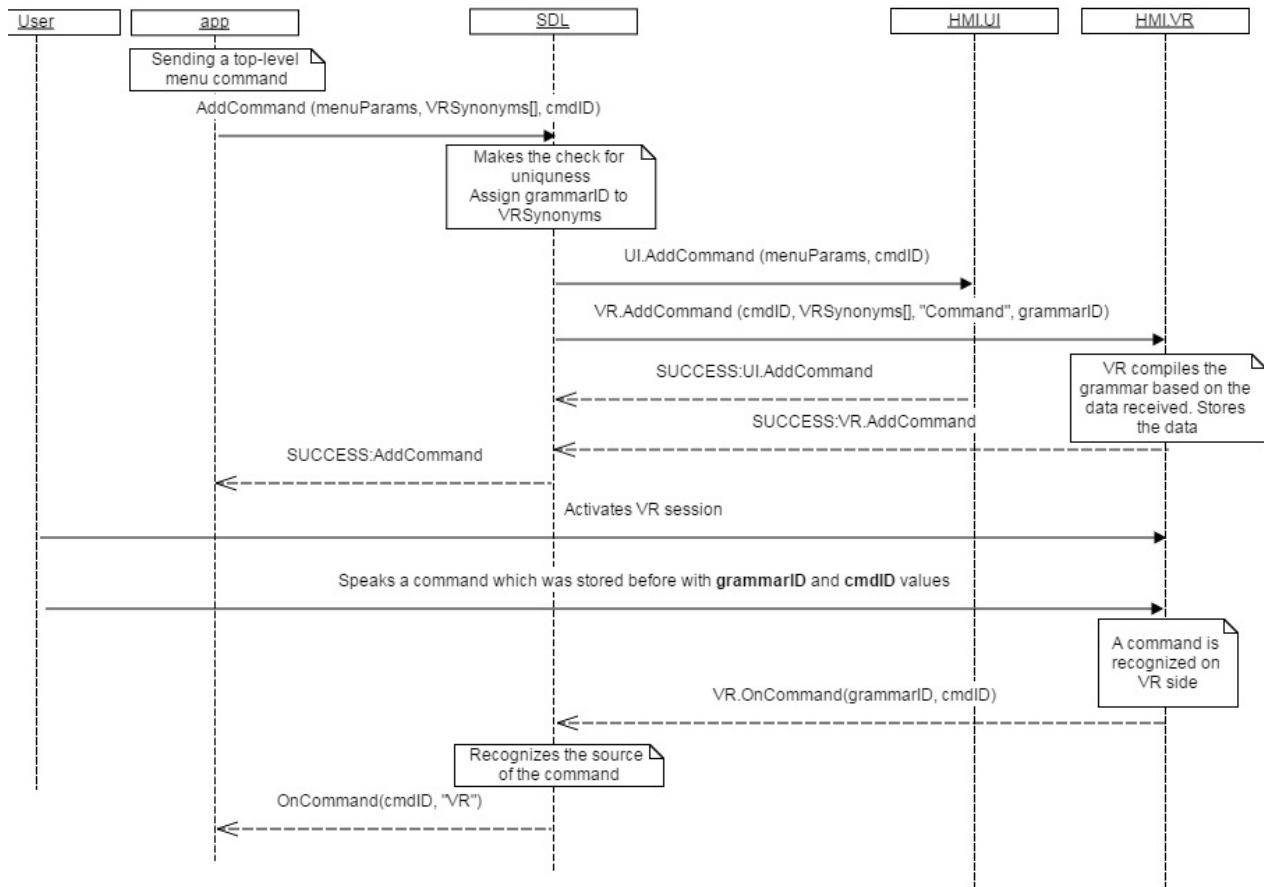
Note: *grammarID is a unique identifier for compiled VR grammar for a command or a set of commands (e.g. a command and its synonyms). So in the VR engine's database, there may be multiple cmdID entries under each unique grammarID. grammarID is assigned on SDL's side at the time of sending [VR.AddCommand](#) request to HMI.*

9.10.1.1 Parameters

Param Name	Type	Mandatory	Additional	Description
cmdID	Integer	true		The ID of the command selected by the User.
grammarID	Integer	true	minvalue=0 maxvalue=20000 00000	ID of the specific grammar that the command relates to. Previously provided by SDL via corresponding VR.AddCommand request.
appID	Integer	false		ID of the application that relates to this RPC.

9.10.2 Sequence Diagrams

9.10.2.1 VR.OnCommand



9.10.3 JSON Messages Examples

```
{
    "jsonrpc" : "2.0",
    "method" : "VR.OnCommand",
    "params" :
    {
        "cmdID" : 4365,
        "grammarID" : 11,
        "appID" : 12564
    }
}
```

9.11 OnLanguageChange

9.11.1 Description

Type:	Notification
Sender:	HMI
Purpose:	Inform about language of VR engine is changed.

SDL needs to be in the know when the User changes the language of voice recognition: Upon the receipt of `OnLanguageChange` notification SDL will unregister the applications of different language to provide them with possibility to re-register with the correct (new HMI) VR language.

HMI must:

- 1) Send the `VR.OnLanguageChange` notification when the User switches VR to another language and provide this new value via `language` parameter.

Note: SDL ignores all invalid notifications which come from HMI (invalid JSON, invalid data types/bounds etc)
In case of several applications registered with HMI, upon `OnLanguageChange` receipt SDL will send `OnAppUnregistered` notifications for each of appIDs with VR language different to new one.

After the applications re-register with the new language (provided via `OnLanguageChange`), SDL will send the corresponding `OnAppRegistered` notifications to HMI.

9.11.1.1 Parameters

Param Name	Type	Mandatory	Description
language	Common.Language	true	Language VR has been switched to. See Language

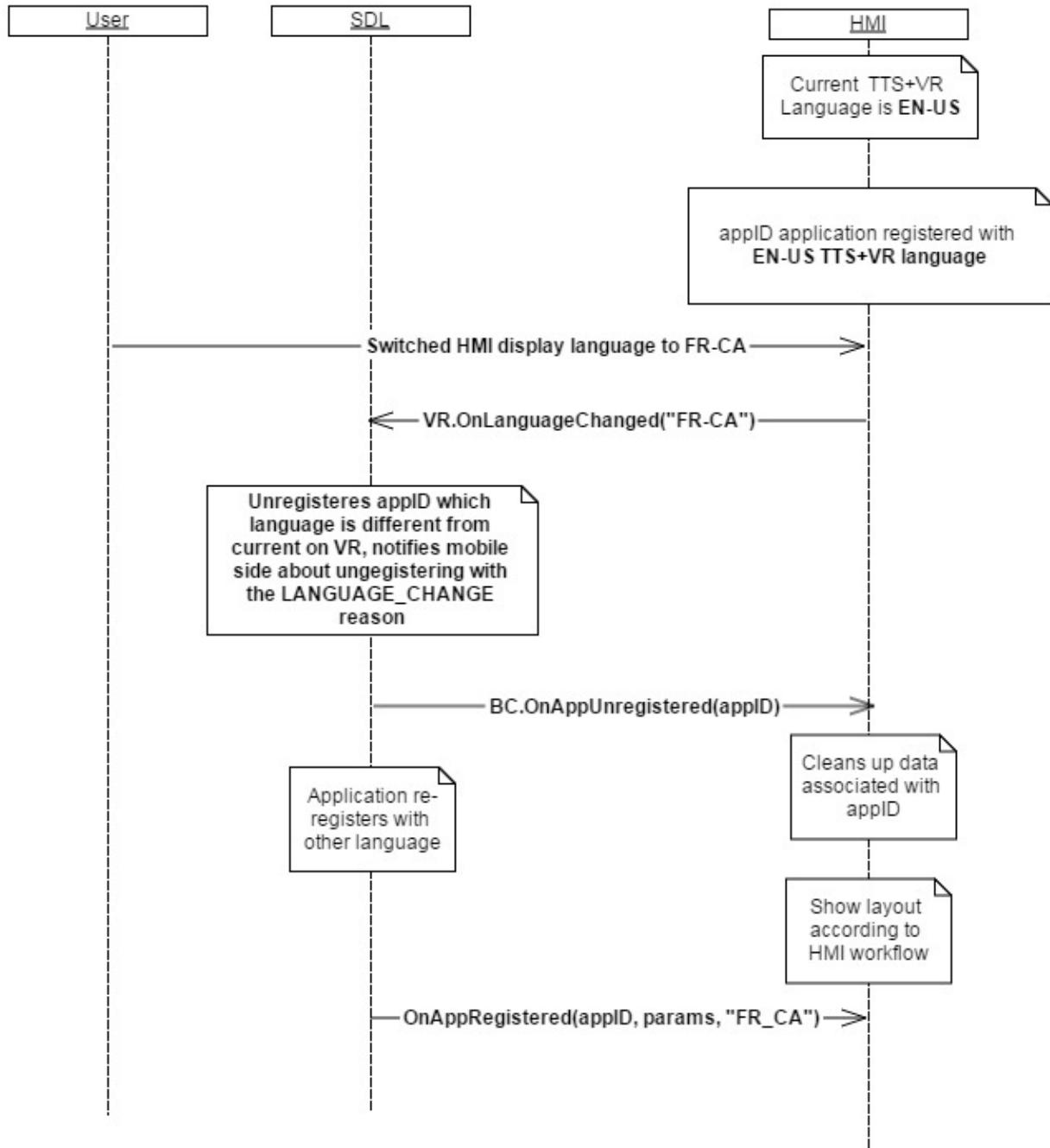
9.11.1.2 Language

Element 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

Element Name	Value	Description
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
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 Belgium (Flemish)
EL-GR	25	Greek
HU-HU	26	Hungarian
FI-FI	27	Finnish
SK-SK	28	Slovak

9.11.2 Sequence Diagrams

9.11.2.1 VR.OnLanguageChange



9.11.3 JSON Messages Examples

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

9.12 VR.PerformInteraction

9.12.1 Description

Type:	Function
Sender:	SDL
Purpose:	Perform a VR interaction with the User.

SDL uses PerformInteraction for requesting an interaction with the User, for example:

- To confirm a command
- To make a choice between several suggested one items in a list

The request concerns the application currently active on HMI.

Notes about PerformInteraction

1. *SDL may intend to perform one of three modes of interaction:*
 - 1.1. *'Manual only' – when the User is expected to make a choice manually only (display, buttons, soft buttons are involved).*
 - 1.2. *'VR only' – when the User is expected to make a choice by voice only (voice recognition module is involved).*
 - 1.3. *'Both' – when the User is expected to make a choice whichever by voice or manually (display, buttons, soft buttons, VR module are involved).*
2. *Important: SDL may send both VR.PerformInteraction and UI.PerformInteraction for each of above types of interaction. HMI's expected behavior differs depending on interaction mode (see section 9.12.2.1 Behavior).*
3. *SDL provides HMI with the set(s) of choices to be presented to the User. See the below Notes of CreateInteractionChoiceSet.*

Notes about CreateInteractionChoiceSet:

1. *Mobile application in advance creates a set (or several sets) of choices on SDL to be further used in PerformInteraction.*
2. *SDL in advance provides HMI with VR synonyms of created choices via VR.AddCommand (see section 9.4 for details). They are then referenced in VR.PerformInteraction (see section 9.12).*
3. *SDL sends UI choices (those to be displayed) within UI.PerformInteraction directly.*

9.12.2 Request

9.12.2.1 Behavior

HMI must:

1. Distinguish the interaction mode by the following characteristics:

Mode	SDL sends both RPCs with the following parameters:	Note
------	--	------

VR_ONLY	VR.PerformInteraction {grammarID, params}	grammarID is present
	UI.PerformInteraction {params}	No choiceSet
MANUAL_ONLY	VR.PerformInteraction {params}	No grammarID
	UI.PerformInteraction {choiceSet, params}	choiceSet is present
BOTH	VR.PerformInteraction {grammarID, params}	grammarID is present
	UI.PerformInteraction {choiceSet, params}	choiceSet is present

I. 'VR only' interaction mode:

HMI must:

1. Store the provided text values of initial, help and timeout prompts (see below for how to use) as well as the value of timeout.
2. Start VR session using VR commands of the corresponding grammarID with HMI-defined timeout.
3. Speak the `initialPrompt` right after request is received (sending notifications of `TTS.Started` and `TTS.Stopped` before and after speaking the prompt correspondingly).
4. Start counting down the value provided within `timeout` parameter (when ran out `timeoutPrompt` must be spoken). General timeout of the whole request depends on HMI-defined VR session timeout.
5. Speak the `timeoutPrompt` (sending notifications of `TTS.Started` and `TTS.Stopped` before and after speaking the prompt correspondingly) when `timeout` value runs out.
6. Respond the request with one of applicable result codes upon:
 - HMI-defined-VR-session timeout (ABORTED result)
 - User's choice (SUCCESS result) providing the `choiceID` (that is equal to the `cmdID` of corresponding `VR.AddCommand`)

Forget the `prompts` data after the interaction completes

II. 'Manual only' interaction mode:

1. Store the provided text values of initial, help and timeout prompts (see below for how to use) as well as the value of timeout.

2. Respond the request with `SUCCESS` result codes
(for all of applicable result codes please see section 9.12.3 Response).
3. Speak the `initialPrompt` right after request is received (sending notifications of `TTS.Started` and `TTS.Stopped` before and after speaking the prompt correspondingly).
4. Start counting down the value provided within `timeout` parameter.
5. Speak the `timeoutPrompt` (sending notifications of `TTS.Started` and `TTS.Stopped` before and after speaking the prompt correspondingly) when `timeout` value runs out.
6. Forget the prompts data upon the corresponding `UI.PerformInteraction` is completed by user's UI choice or aborted by the user or some event.

III. 'Both' interaction mode:

1. Store the provided text values of initial, help and timeout prompts (see below for how to use) as well as the value of timeout.
2. Start VR session using VR commands of the corresponding grammarID with HMI-defined timeout.
3. Speak the `initialPrompt` right after request is received (sending notifications of `TTS.Started` and `TTS.Stopped` before and after speaking the prompt correspondingly).
4. Start counting down the value provided within `timeout` parameter.
5. Speak the `timeoutPrompt` (sending notifications of `TTS.Started` and `TTS.Stopped` before and after speaking the prompt correspondingly) when `timeout` value runs out.
6. Respond the request with one of applicable result codes upon:
 - HMI-defined-VR-session timeout (`ABORTED` result) AND proceed with **step 7**.
 - User's choice (`SUCCESS` result) providing the `choiceID` (that is equal to the `cmdID` of corresponding `VR.AddCommand`)
7. Repeat step 4. -5. for corresponding `UI.PerformInteraction` and forget the prompts values when `UI.PerformInteraction` completes.

9.12.2.2 Parameters

Param name	Type	Mandatory	Additional	Description
helpPrompt	Common.TTSChunk	false	Array = true minsize = 1 maxsize = 100	This is the array of strings to be spoken when the HMI-defined 'Help' VR command is recognized during PerformInteraction (see TTSChunk).
initialPrompt	Common.TTSChunk	true	Array = true minsize = 1 maxsize = 100	The initial prompt that must be spoken to the User right after request comes to HMI (see TTSChunk).
timeoutPrompt	Common.TTSChunk	false	Array = true minsize = 1 maxsize = 100	Must be spoken upon timeout defined within request (see TTSChunk).
timeout	Integer	true	-	Timeout. When ran out timeoutPrompt must be spoken.
grammarID	Integer	false	Array = true minsize = 1 maxsize = 100 minvalue = 0 maxvalue = 2000000000	ID of VR choice set: contains a set of commands previously added via VR.AddCommand that have one and the same grammarID.

9.12.3 Response

Note:

There is a difference in message type for WebSocket and D-Bus connection (see the table below).

Result	Description	Message type		Message Params	Notes
		WebSocket	D-Bus		
Success	SUCCESS The cases described in 7.12.2: '1.2', 'II.3', 'III.3', 'III.4.3'.	JSON response	Regular response	choiceID, code: 0	See section 9.12.2 Parameters.
Failure	ABORTED The interaction is aborted by the User or system event of higher priority.	JSON error message	Regular response	code: 5	Applicable for this RPC result codes. Please see Result Enumeration for all SDL-supported codes.
	TIMED_OUT The User has not made a choice during HMI-defined timeout.			code: 10	
	INVALID_ID Wrong appID or grammarID received			code: 13	
	GENERIC_ERROR 1) The unknown issue occurred or other codes are not applicable.			code: 22	

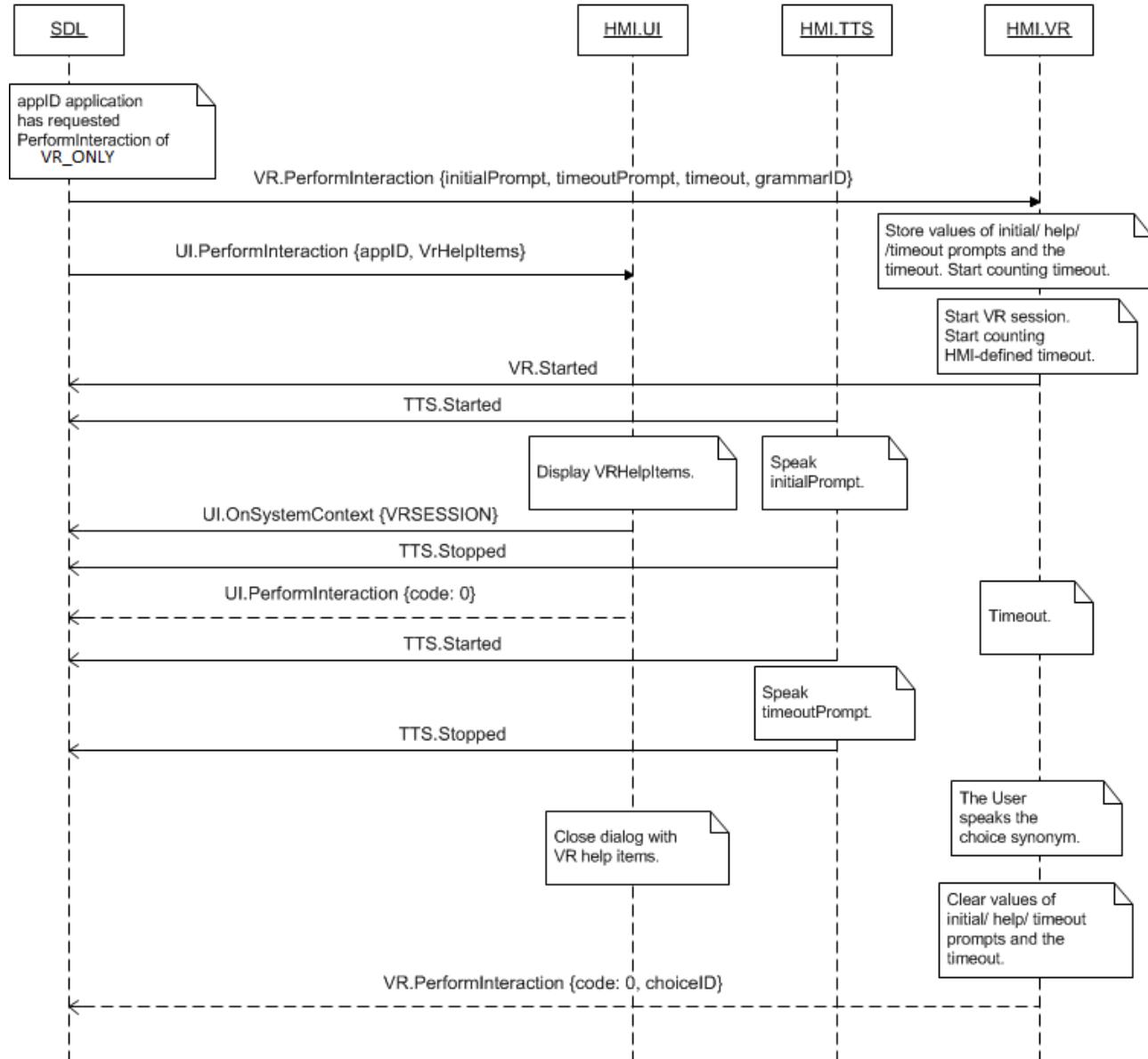
9.12.4 Parameters

Param Name	Type	Mandatory	Additional	Description
------------	------	-----------	------------	-------------

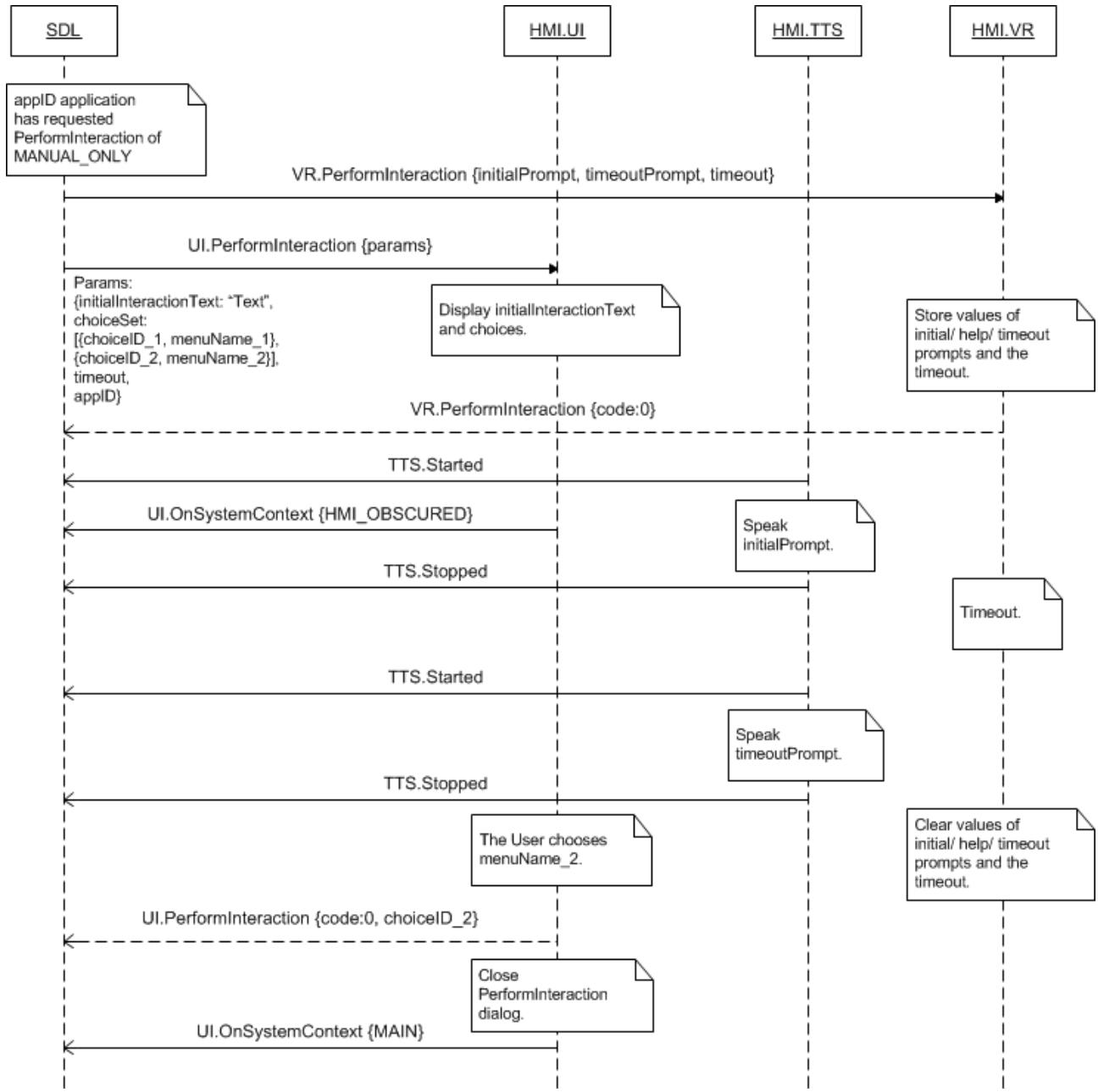
Param Name	Type	Mandatory	Additional	Description
choiceID	Integer	false	Minvalue = 0 Maxvalue = 2000000000	ID that represents the choice made by the User (one among those provided within the request). Equals to cmdID of the corresponding VR.AddCommand

9.12.4 Sequence Diagrams

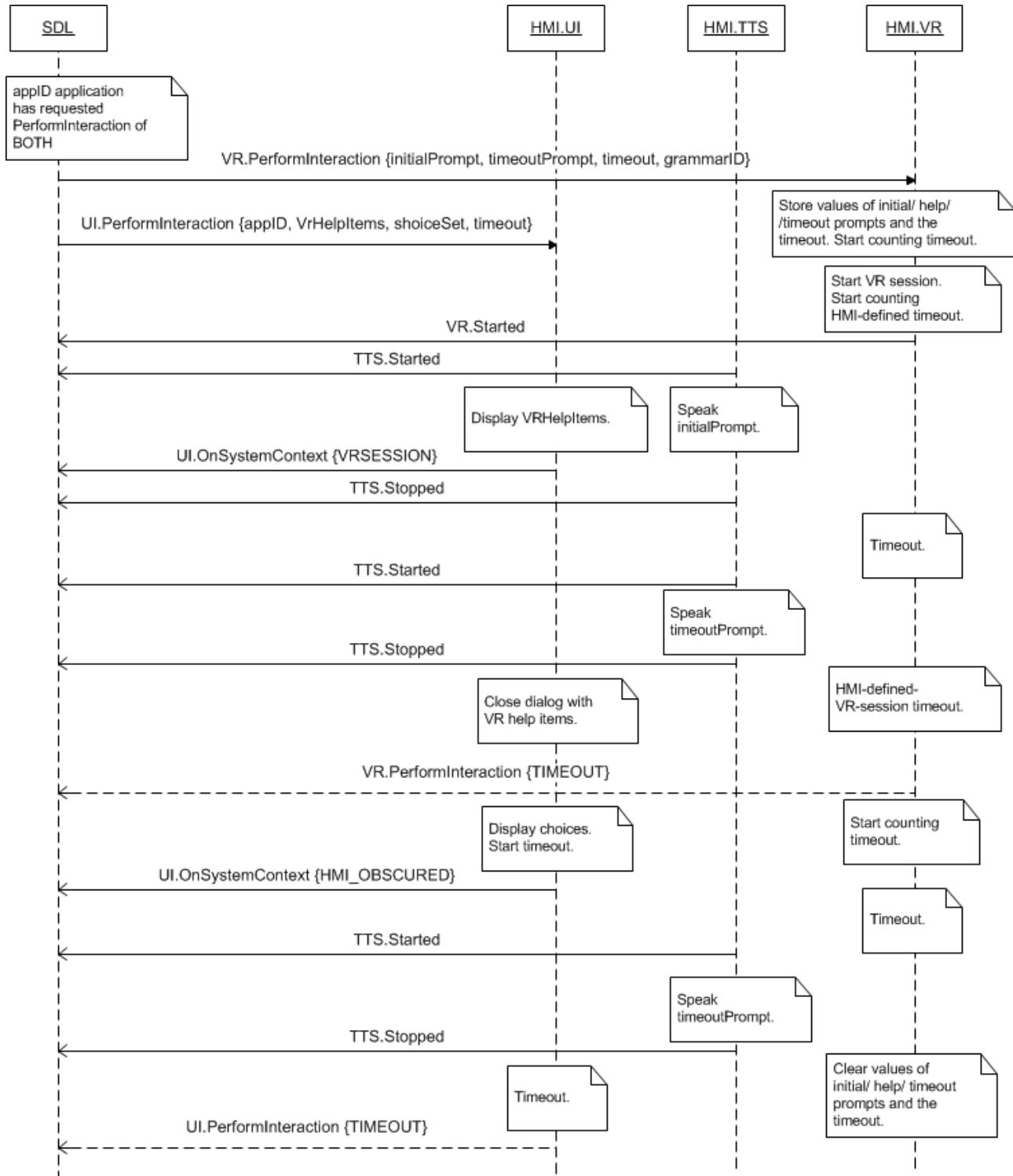
9.12.4.1 VR.PerformInteraction in ‘VR only’ mode successfully completed



9.12.4.2 VR.PerformInteraction in ‘Manual only’ mode successfully completed



9.12.4.3 VR.PerfromInteraction in 'Both' mode timed out



9.12.5 JSON Messages Examples

9.12.5.1 Request

```
{
    "id" : 79,
    "jsonrpc" : "2.0",
    "method" : "VR.PerformInteraction",
    "params" :
    {
        "initialPrompt": "Please enter your name.",
        "timeoutPrompt": "Time is up! Please respond.",
        "timeout": 10,
        "grammarID": "NAME_GRAMMAR"
    }
}
```

```

    "initialPrompt" :
    [
        {
            "text" : "Please make your choice by voice",
        },
        {
            "text" : "Yes",
            "text" : "No",
            "text" : "Skip"
        },
        {
            "text" : "The time is about to run out"
        }
    ],
    "timeoutPrompt" :
    [
        {
            "text" : "The time is about to run out"
        }
    ],
    "timeout" : 10000,
    "grammarID" : 245
}
}

```

9.12.5.2 Response

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

9.12.5.3 Error message

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

10 TTS Component Description

10.1 IsReady

10.1.1 Description

Type:	Function
Sender:	SDL
Purpose:	Get information if Text-To-Speech module is ready to communicate with SDL.

The request comes after HMI's readiness is confirmed via [OnReady](#) notification. SDL requires the information about whether the TTS module is physically present on HU and if so whether it is ready to communicate with SDL.

Note:

If TTS module is responded to be unavailable, SDL will not further send the requests related to it.

10.1.2 Request

10.1.2.1 Behavior

HMI must:

- 1) Check whether TTS module is present and ready to communicate with SDL
- 2) Respond correspondingly to results of this check

10.1.3 Response

Note:

There is a difference in message type for WebSocket and D-Bus connection (see the table below).

Result	Description	Message type		Message Params	Notes
		WebSocket	D-Bus		
Success	SUCCESS: HMI provides the information about TTS availability.	JSON response	Method return	available, code: 0	
Failure	INVALID_DATA: The data sent is invalid (invalid JSON syntax)	JSON error message	Method return	code: 11	Applicable for this RPC result codes. Please see Result Enumeration for all SDL-supported codes.
	DATA_NOT_AVAILABLE: The information about TTS availability cannot be provided.			Code: 9	
	GENERIC_ERROR: The unknown issue occurred or other codes are not applicable.			code: 22	

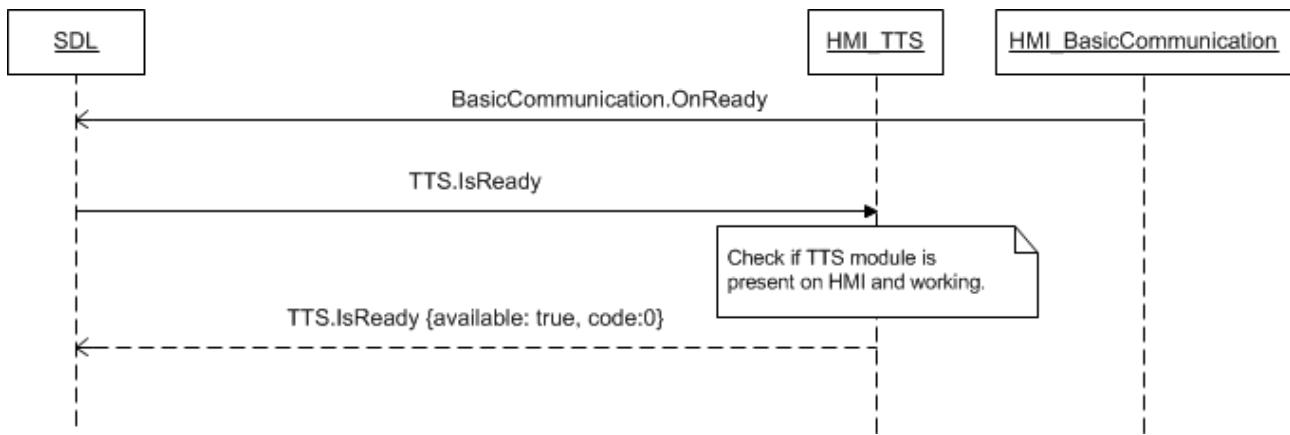
10.1.3.1 Parameters

Param Name	Type	Mandatory	Description
------------	------	-----------	-------------

availabe	Boolean	true	Must be - 'true' if UI is present and ready - 'false' if not.
----------	---------	------	---

10.1.4 Sequence Diagrams

10.1.4.1 TTS.IsReady and preceding OnReady



10.1.5 JSON Messages Examples

10.1.5.1 Request

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

10.1.5.2 Response

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

10.1.5.3 Error message

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

```

        "method" : "TTS.IsReady"
    }
}
}

```

10.2 GetCapabilities

10.2.1 Description

Type:	Function
Sender:	SDL
Purpose:	Get Text-To-speech module capabilities

SDL can store the HMI capabilities in `smartDeviceLink.ini` file saved in SDL directory (see `HMICapabilities` parameter of ini file). Nevertheless, on its startup after getting `OnReady` notification and response to `TTS.IsReady` from HMI, SDL sends this RPC for obtaining the capabilities of TTS.

SDL Note: In case SDL didn't get TTS capabilities from HMI, it will use the default ones from the `hmi_capabilities.json` file. The path to the file is defined in `smartDeviceLink.ini` as `HMICapabilities` parameter. See also [15.3.2.3 TTS section of hmi_capabilities.json](#)

10.2.2 Request

7.1.2.1 Behavior

HMI must:

- 1) Check the TTS capabilities.
- 2) Respond correspondingly to results of this check.

10.2.3 Response

Note:

There is a difference in message type for WebSocket and D-Bus connection (see the table below).

Result	Description	Message type		Message Params	Notes
		WebSocket	D-Bus		
Success	SUCCESS: HMI provides the TTS capabilities.	JSON response	Method return	capabilities, prerecordedSpeechCapabilities, code: 0	
Failure	INVALID_DATA: The data sent is invalid (invalid JSON syntax)	JSON error message	Method return	code: 11	Applicable for this RPC result codes.
	DATA_NOT_AVAILABLE: The TTS capabilities cannot be provided.			Code: 9	Please see Result Enumeration for all SDL-supported codes.

	GENERIC_ERROR: The unknown issue occurred or other codes are not applicable.			code: 22	
--	---	--	--	----------	--

10.2.3.1 Parameters

Param name	Type	Mandatory	Additional	Description
capabilities	Common.Speech Capabilities	true	array = true minsize = 1 maxsize = 5	Contains information about the TTS capabilities.
prerecordedSpeechCapabilities	Common.PrerecordedSpeech	true	array = true minsize = 1 maxsize = 5	Contains a list of prerecorded speech items present on the platform.

10.2.3.2 SpeechCapabilities

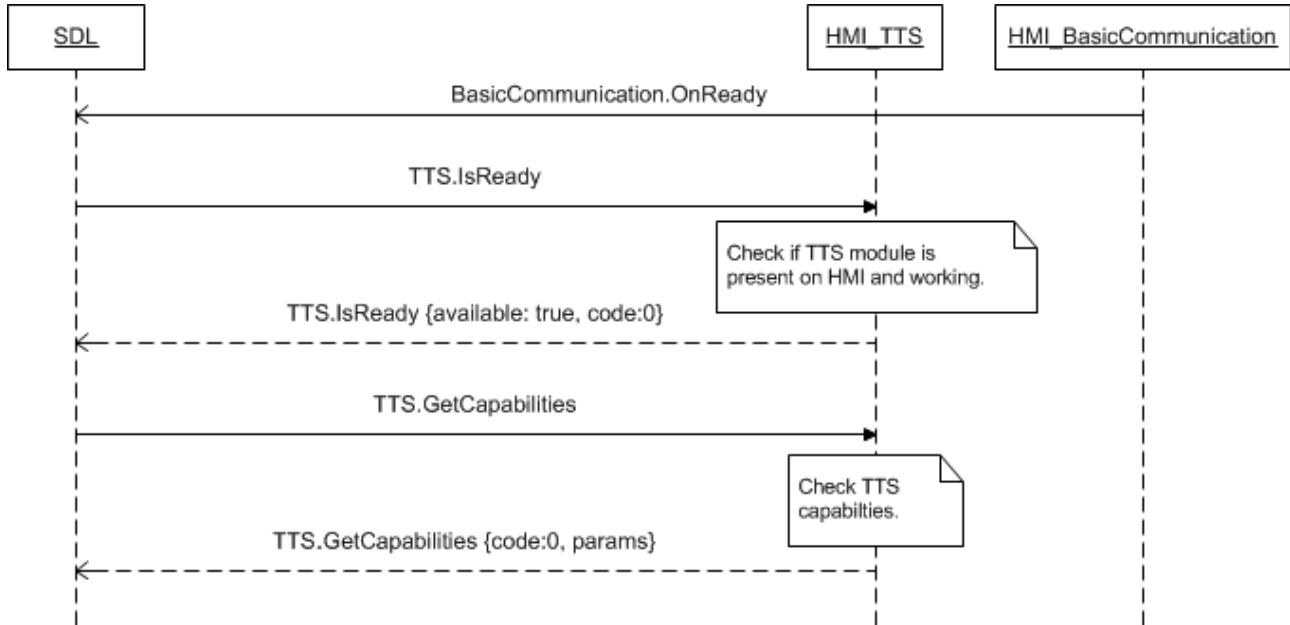
Element name	Value	Short Description
TEXT	0	HMI has the capability to speak the provided text.
SAPI_PHONEMES	1	Speech API phonemes: used for creating pronunciations for words that are not currently in the lexicon.
LHPLUS_PHONEMES	2	LH Plus API phonemes: used for creating pronunciations for words that are not currently in the lexicon.
PRE_RECORDED	3	HMI has the capability to speak the phrases pre-recorded on HMI.
SILENCE	4	HMI has the capability to play silence over TTS during the default HMI-defined period of time.

10.2.3.3 PrerecordedSpeech

Element name	Value	Short Description
HELP_JINGLE	0	Pre-recorded help phrase on HMI
INITIAL_JINGLE	1	
LISTEN_JINGLE	2	
POSITIVE_JINGLE	3	
NEGATIVE_JINGLE	4	

10.2.4 Sequence Diagrams

10.1.4.1 TTS.GetCapabilities



10.2.5 JSON Messages Examples

10.2.5.1 Request

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

10.2.5.2 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"
    }
}
```

10.2.5.3 Error message

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

```

    "data" :
    {
        "method" :
    "TTS.GetCapabilities"
    }
}

```

10.3 GetSupportedLanguages

10.3.1 Description

Type:	Function
Sender:	SDL
Purpose:	Get languages supported by Text-To-speech module

Once TTS module is confirmed to be ready (via response to IsReady RPC) SDL starts discovering its capabilities via GetCapabilities and GetSupportedLanguages RPCs.

Response to TTS.GetSupportedLanguages is assumed to bring the information about what languages are supported for pronouncing by text-to-speech module. Having obtained this information SDL will monitor the language parameter within RPCs from mobile application(s) and reject the requests containing language not supported by HMI.

Note:

The list of languages recognized by SDL is provided in the section 10.3.3.2 Language Enumeration.

10.3.2 Request

7.1.2.1 Behavior

HMI must:

- 1) Check the TTS supported languages
- 2) Respond correspondingly to results of this check.

10.3.3 Response

Note:

There is a difference in message type for WebSocket and D-Bus connection (see the table below).

Result	Description	Message type		Message Params	Notes
		WebSocket	D-Bus		
Success	SUCCESS: HMI provides the TTS supported languages.	JSON response	Method return	languages, code: 0	
Failure	INVALID_DATA: The data sent is invalid (invalid JSON syntax)	JSON error message	Method return	code: 11	Applicable for this RPC result codes.

	DATA_NOT_AVAILABLE: The TTS supported languages cannot be provided.		Code: 9	Please see Result Enumeration for all SDL-supported codes.
	GENERIC_ERROR: The unknown issue occurred or other codes are not applicable.		code: 22	

10.3.3.1 Parameters

Param name	Type	Mandatory	Additional	Description
languages	Common.Language	true	Array = true minsize = 1 maxsize = 100	List of languages supported in TTS. See Language.

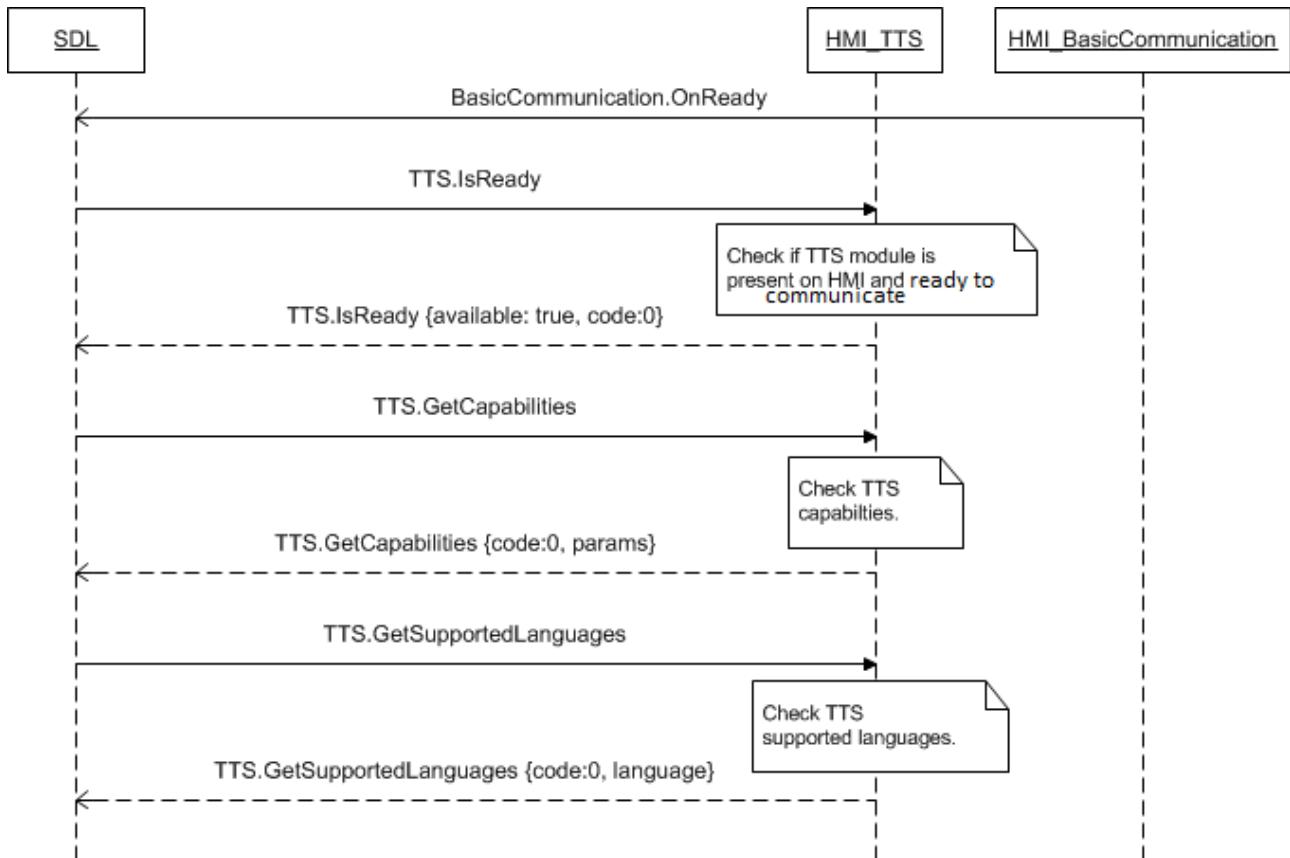
10.3.3.2 Language

Element 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
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 Belgium (Flemish)

Element Name	Value	Description
EL-GR	25	Greek
HU-HU	26	Hungarian
FI-FI	27	Finnish
SK-SK	28	Slovak

10.3.4 Sequence Diagrams

10.3.4.1 TTS.GetSupportedLanguages



10.3.5 JSON Messages Examples

10.3.5.1 Request

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

10.3.5.2 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"
}
}
```

10.3.5.3 Error message

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

10.4 Speak

10.4.1 Description

Type:	Function
Sender:	SDL
Purpose:	Prompt TTS to speak the requested text

SDL provides the text information for HMI to speak it to the User via TTS engine.

10.4.2 Request

10.4.2.1 Behavior

HMI must:

1. Notify SDL via TTS.Started about speaking is started
2. Depending on HMI capabilities, attenuate or make the audio (if streamed by the application or HMI itself) not audible.
3. Distinguish the type of request (please see section [10.4.2.5 MethodName](#)):
 - 3.1. ALERT-type requests have the highest priority: HMI must speak the requested text by all means and right away
 - 3.2. SPEAK type requests have the normal priority: HMI must speak the requested text, still the delays are acceptable.
 - 3.3. AUDIO_PASS_THRU has normal priority type on HMI. HMI must speak the requested text, still the delays are acceptable (see [7.20.4.1](#))

- [PerformAudioPassThru](#) diagram for more details).
- 3.4. ALERT_MANEUVER has normal priority type on HMI. HMI must speak the requested text, still the delays are acceptable (see [12.2.4 AlertManeuver](#) diagram for more details).
 - 3.5. If type is omitted within request, HMI must speak the requested text; still the negative response is acceptable.
 4. Speak the text requested within TTSShunk parameter. The text value is specified together with 'type' parameter that defines how exactly this text must be used by HMI:
 - TEXT – the text is provided and must be pronounced by HMI (letter-to-sound).
 - SAPI_PHONEMES / LHPLUS_PHONEMES – the phonemes are provided and must be pronounced by HMI.
 - PRE_RECORDED – the provided text must be pronounced by pre-recorded phrases.
 - SILENCE – HMI must play silence over TTS during the default HMI-defined period of time.
 5. Respond the request with successful result code (the cases of non-successful results are described in section 10.4.3 Response).
 6. Notify SDL via TTS.Stopped about speaking is finished.

10.4.2.2 Parameters

Param name	Type	Mandatory	Additional	Description
ttsChunks	Common.TTSC hunk	true	Array = true minsize = 1 maxsize = 100	List of strings to be spoken. See TTSCunks
speakType	Common. MethodName	false	-	Defines the type of TTS.Speak request
appID	Integer	false	-	ID of the application that requested this RPC.

10.4.2.3 TTSShunk

Param name	Type	Mandatory	Additional	Description
text	String	true	Maxlength = 500	The text or phonemes to speak
type	Common.SpeechCa pabilities	true	-	Describes, whether it is text or a specific phoneme set. See SpeechCapabilities.

10.4.2.4 SpeechCapabilities

Element name	Value	Short Description
TEXT	0	Text is provided to HMI to be spoken.
SAPI_PHONEMES	1	Speech API phonemes are provided and must be used by HMI for pronouncing.

Element name	Value	Short Description
LHPLUS_PHONEMES	2	LH Plus phonemes are provided and must be used by HMI for pronouncing.
PRE_RECORDED	3	The text is provided and must be spoken as a pre-recorded phrase by HMI.
SILENCE	4	HMI must play silence over TTS during the default HMI-defined period of time.

10.4.2.5 MethodName

Element name	Value	Short Description
ALERT	0	TTS.Speak has the <u>alert type</u> (that is, either accompanies UI.Alert RPC or is requested independently from UI as an alert TTS prompting to the User)
SPEAK	1	TTS.Speak has the 'regular' speak type (that is, a friendly reminding or notification or informing, etc., by TTS to the User).
AUDIO_PASS_THRU	2	TTS.Speak has this type when it prompts to the User as a part of PerformAudioPassThru. See 7.20.4.1 PerformAudioPassThru diagram for more details
ALERT_MANEUVER	3	Defines the type of the mobile API request (AlertMeneuver mobile API) which initiates the request on HMI e.g. (TTS.Speak, BC.PlayTone)

10.4.3 Response

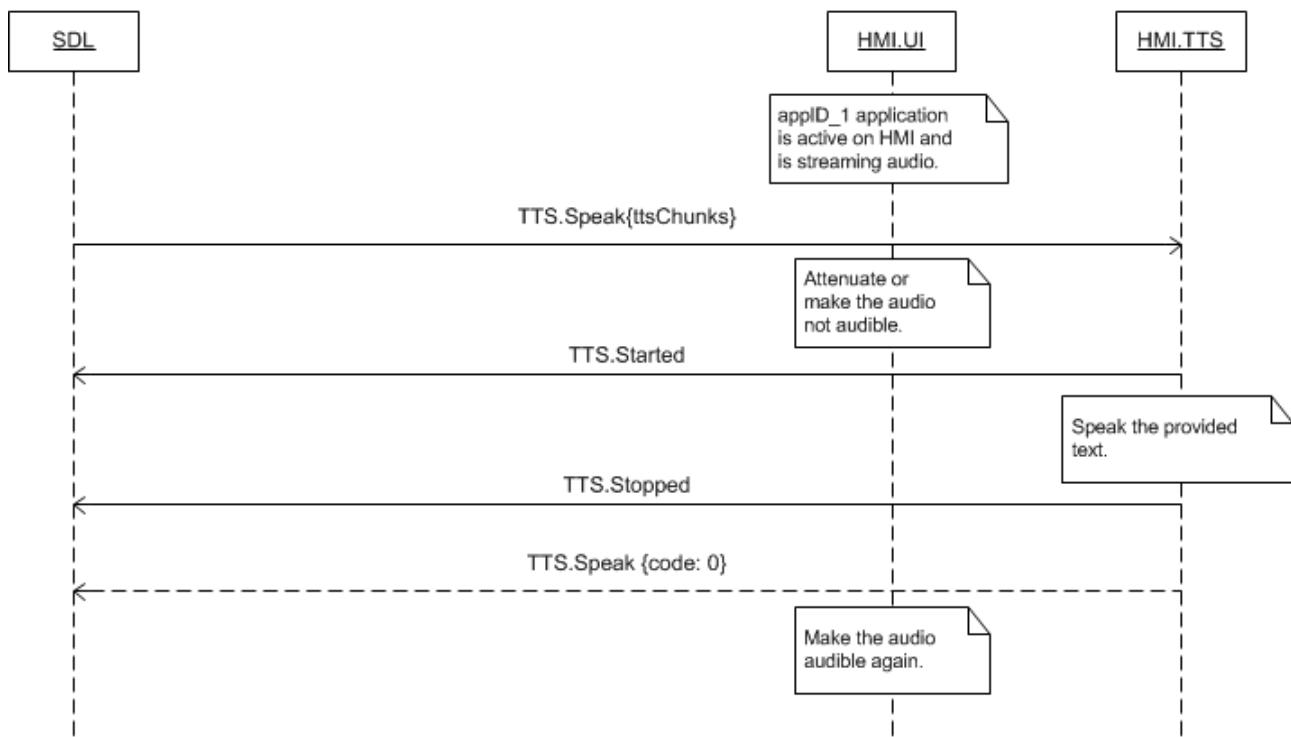
Note:

There is a difference in message type for WebSocket and D-Bus connection (see the table below).

Result	Description	Message type		Message Params	Notes
		WebSocket	D-Bus		
Success	SUCCESS: HMI speaks the requested text and responds when finished.	JSON response	Method return	code: 0	
Failure	INVALID_DATA: The data sent is invalid (invalid JSON syntax, parameters are out of bounds or of wrong type)	JSON error message	Method return	code: 11	Applicable for this RPC result codes.
	INVALID_ID Wrong appID is sent			code: 13	
	ABORTED: RPC is interrupted by: 1) StopSpeaking RPC 2) VR session started 3) Another RPC of higher priority			Code: 5	Please see Result Enumeration for all SDL-supported codes.
	GENERIC_ERROR: The unknown issue occurred or other codes are not applicable.			code: 22	

10.4.4 Sequence Diagrams

10.4.4.1 Speak



10.4.5 JSON Messages Examples

10.4.5.1 Request

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

10.4.5.2 Response

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

10.4.5.3 Error message

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

10.5 StopSpeaking

10.5.1 Description

Type:	Function
Sender:	SDL
Purpose:	Interrupt the Speak RPC.

SDL sends StopSpeaking RPC to abort TTS prompting initiated by TTS.Speak. This RPC is not intended to interrupt HMI-initiated TTS speaking.

10.5.2 Request

10.5.2.1 Behavior

HMI must:

- 1) Stop speaking the text requested via TTS.Speak.
- 2) Respond to TTS.StopSpeaking
- 3) Respond with ABORTED result code for TTS.Speak.
- 4) Notify SDL about TTS has finished speaking via TTS.Stopped.

10.5.3 Response

Note:

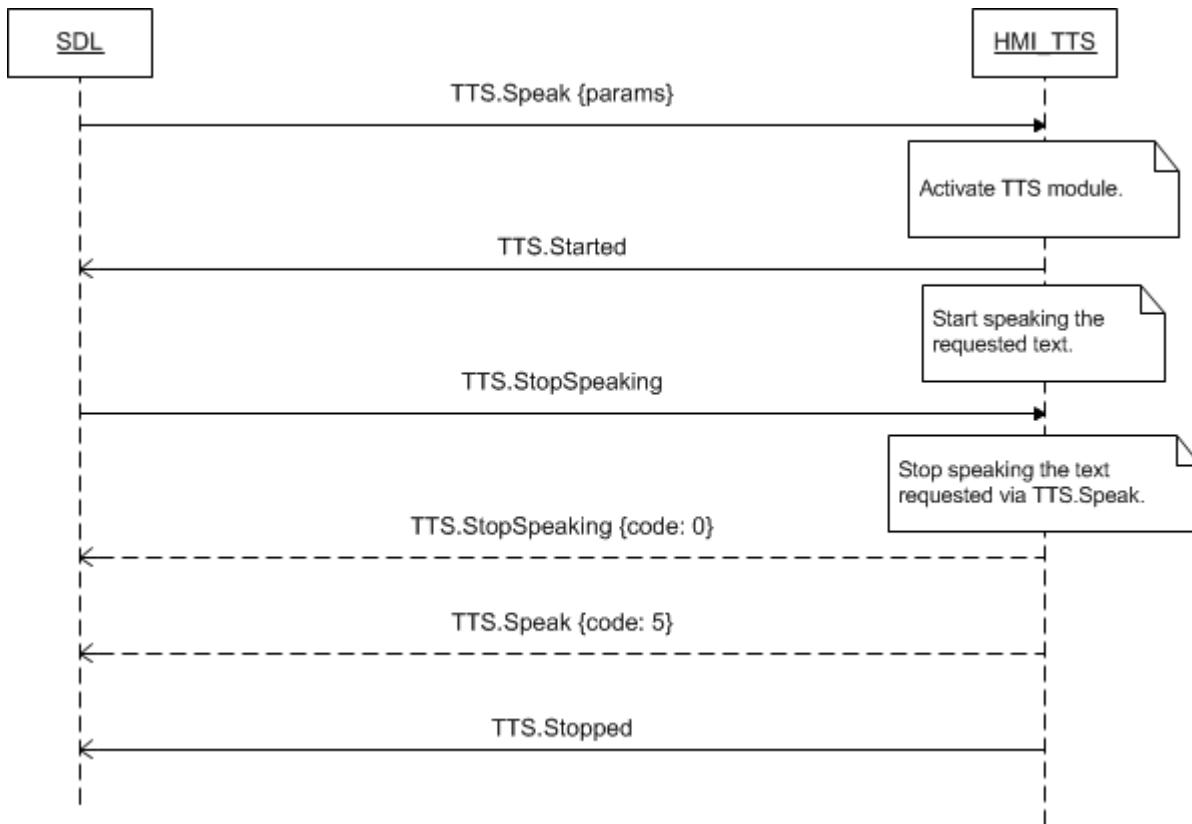
There is a difference in message type for WebSocket and D-Bus connection (see the table below).

Result	Description	Message type		Message Params	Notes
		WebSocket	D-Bus		
Success	SUCCESS: The Speak RPC is interrupted.	JSON response	Method return	code: 0	
Failure	INVALID_DATA: The data sent is invalid (invalid JSON syntax)	JSON error message	Method return	code: 11	Applicable for this RPC result codes.

	GENERIC_ERROR: The unknown issue occurred or other codes are not applicable.			code: 22	Please see Result Enumeration for all SDL-supported codes.
--	---	--	--	----------	--

10.5.4 Sequence Diagrams

10.5.4.1 StopSpeaking



10.5.5 JSON Messages Examples

10.5.5.1 Request

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

10.5.5.2 Response

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

```
}
```

10.5.5.3 Error message

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

10.6 ChangeRegistration

10.6.1 Description

Type:	Function
Sender:	SDL
Purpose:	Change the application language for TTS

When the application registers and its languageDesired value is different from the current HMI TTS language, this application is provided with the possibility to change the language and re-register just this value. TTS.ChangeRegistration RPC is used for such purpose of re-registering the application with HMI current TTS language.

10.6.2 Request

10.6.2.1 Behavior

HMI must:

- 1) Update the appId application related data previously provided via OnAppRegistered with the new application language desired for TTS.
- 2) Respond to the request.

Note:

There may be cases when the application requests to re-register it with the language different from current HMI TTS language (e.g. TTS.ChangeRegistration (ES-ES) while the current language is EN-US):

- HMI must: confirm it accepts such registration (respond with SUCCESS result code)
- HMI is expected to:
 - Recognize the text provided in the registered language for speaking further.
 - Use the rules and pronunciation of the current HMI TTS language (and all the consequences of using TTS engine in the

wrong language (e.g. in ES-ES while the current is EN-US) are on the application).

10.6.2.2 Parameters

Param Name	Type	Mandatory	Additional	Description
language	Common.Language	true		The language requested to be switched to. See Language.
ttsName	Common.TTSChunk	false	array=true minsize=1 maxsize=100	Request new VR synonyms registration Defines an additional voice recognition command. Must not interfere with any name of previously registered applications(SDL makes check).
appId	Integer	true		ID of the application that relates to this RPC.

10.6.2.3 Language

Element 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
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

Element Name	Value	Description
NL-BE	24	Dutch Belgium (Flemish)
EL-GR	25	Greek
HU-HU	26	Hungarian
FI-FI	27	Finnish
SK-SK	28	Slovak

10.6.3 Response

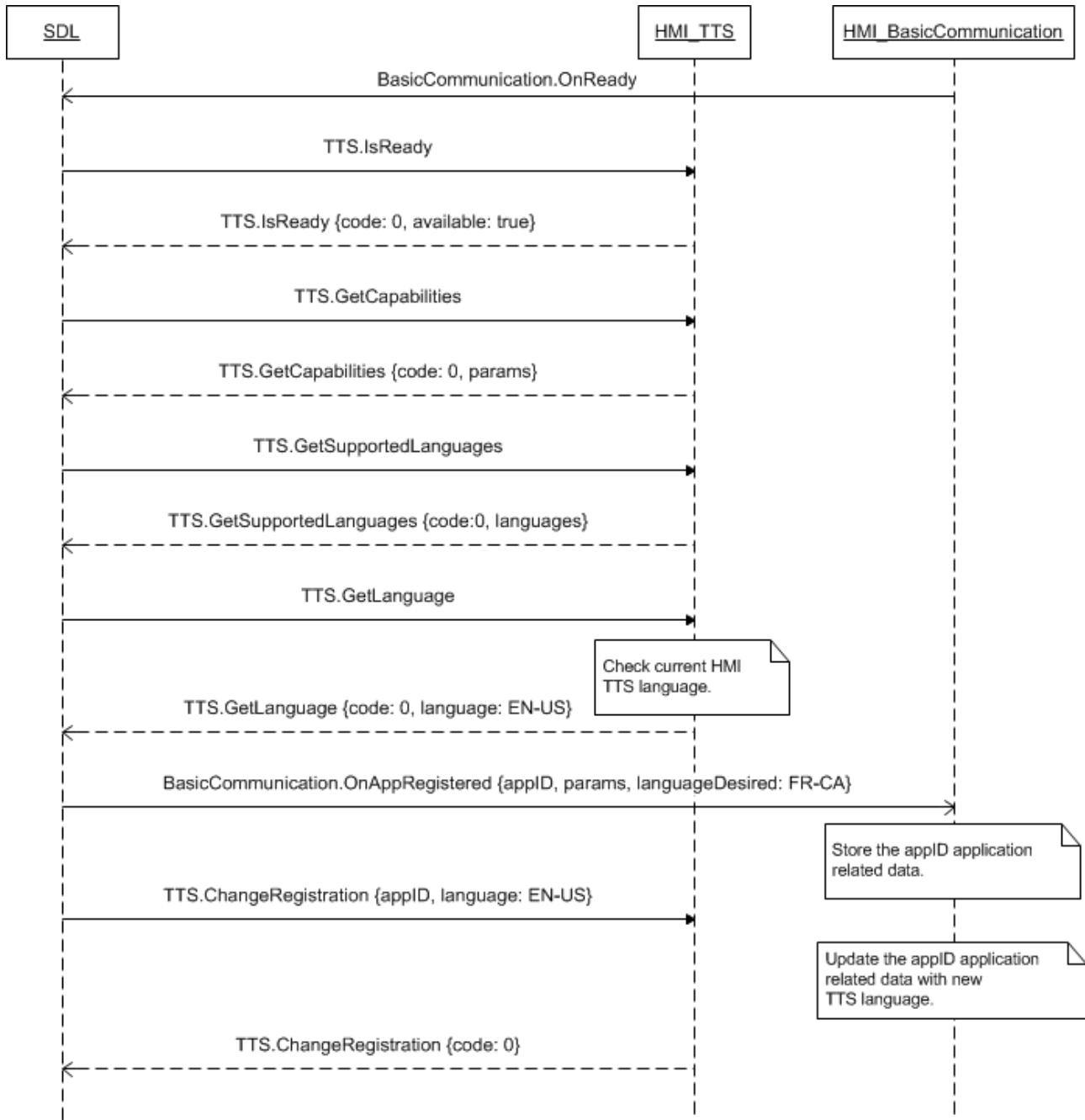
Note:

There is a difference in message type for WebSocket and D-Bus connection (see the table below).

Result	Description	Message type		Message Params	Notes
		WebSocket	D-Bus		
Success	SUCCESS: The requested language has been set for TTS for the named application.	JSON response	Method return	code: 0	
Failure	INVALID_DATA: The data sent is invalid (invalid JSON syntax, parameters are out of bounds or of wrong type)	JSON error message	Method return	code: 11	Applicable for this RPC result codes. Please see Result Enumeration for all SDL-supported codes.
	INVALID_ID Wrong appID is sent			code: 13	
	WRONG_LANGUAGE The language is not supported by TTS			code: 16	
	GENERIC_ERROR: The unknown issue occurred or other codes are not applicable.			code: 22	

10.6.4 Sequence Diagrams

10.6.4.1 TTS.ChangeRegistration after OnAppRegistered



10.6.5 JSON Messages Examples

10.6.5.1 Request

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

10.6.5.2 Response

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

10.6.5.3 Error message

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

10.7 GetLanguage

10.7.1 Description

Type:	Function
Sender:	SDL
Purpose:	Get the current TTS language

SDL inquires the current HMI TTS language.

This RPC is sent by SDL after TTS readiness is confirmed by TTS.IsReady. If later the User changes the HMI language of TTS, HMI must inform SDL about this event via TTS.OnLanguageChange notification.

10.7.2 Request

10.7.2.1 Behavior

HMI must:

- 1) Check the HMI TTS language currently in effect.
- 2) Respond providing SDL with the results of this check.

10.7.3 Response

Note:

There is a difference in message type for WebSocket and D-Bus connection (see the table below).

Result	Description	Message type	Message Params	Notes
--------	-------------	--------------	----------------	-------

		WebSocket	D-Bus		
Success	SUCCESS: HMI provides the language currently active for TTS.	JSON response	Method return	language, code: 0	
Failure	INVALID_DATA: The data sent is invalid (invalid JSON syntax, parameters are out of bounds or of wrong type)	JSON error message	Method return	code: 11	Applicable for this RPC result codes. Please see Result Enumeration for all SDL-supported codes.
	GENERIC_ERROR: The unknown issue occurred or other codes are not applicable.			code: 22	

10.7.3.1 Parameters

Param Name	Type	Mandatory	Description
language	Common.Language	true	The current TTS language. See Language.

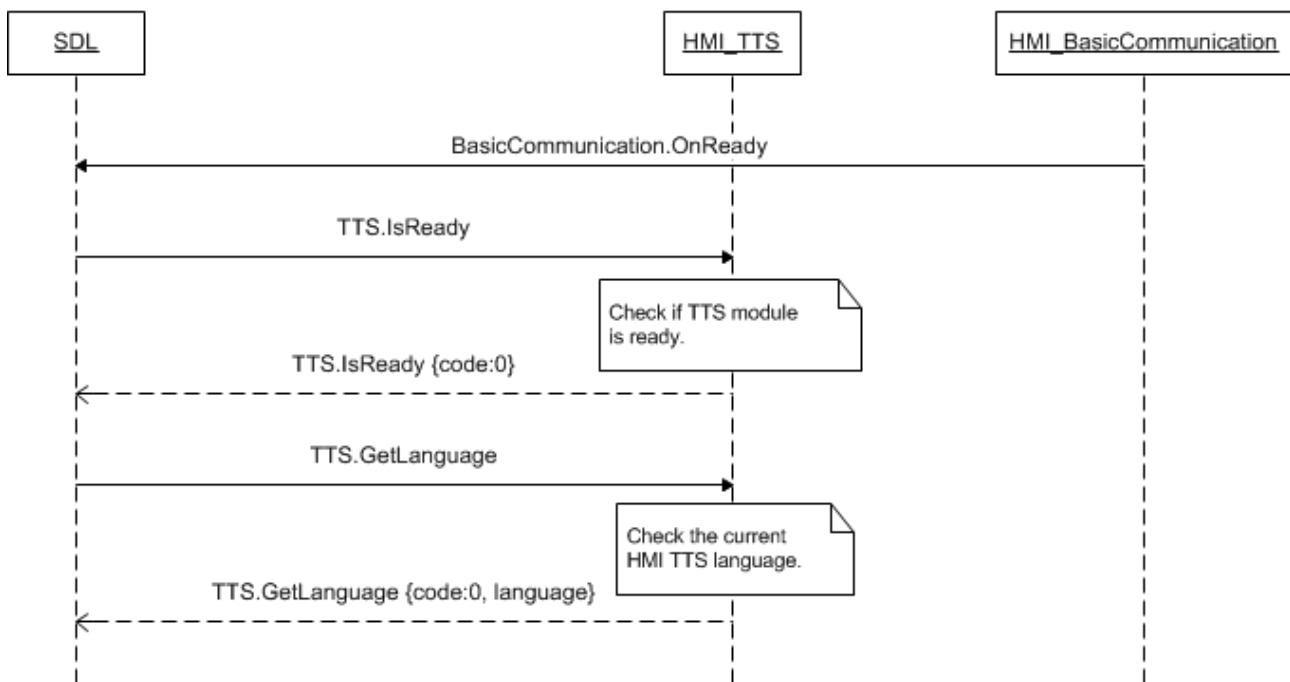
10.6.2.3 Language

Element 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
AR-SA	18	Arabic – Saudi Arabia
KO-KR	19	Korean – South Korea

Element Name	Value	Description
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 Belgium (Flemish)
EL-GR	25	Greek
HU-HU	26	Hungarian
FI-FI	27	Finnish
SK-SK	28	Slovak

10.7.4 Sequence Diagrams

10.7.4.1 TTS.GetLanguage



10.7.5 JSON Messages Examples

10.7.5.1 Request

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

10.7.5.2 Response

```
{
  "id" : 110,
```

```

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

```

10.7.5.3 Error message

```

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

```

10.8 SetGlobalProperties

10.8.1 Description

Type:	Function
Sender:	SDL
Purpose:	Set the properties for TTS

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

10.8.2 Request

10.8.2.1 Behavior

HMI must:

1. Store the provided values associating them with the provided appID.
2. Speak the provided text during the following events related to the appID application:
 - helpPrompt – when the HMI-defined 'Help' command is recognized by VR.
 - timeoutPrompt – when the HMI-defined timeout for VR session is about going to end. After speaking this prompt HMI must provide some additional time for the user to be able to give a command by VR. After this additional amount of time is out, VR session must be closed.

3. Respond with **SUCCESS** result code after the requested values have been stored. All of applicable result codes please see in the section 10.8.3 Response.

10.8.2.2 Parameters

Param name	Type	Mandatory	Additional	Description
helpPrompt	Common.TTSSchunk	false	Array = true minsize = 0 maxsize = 100	This is the array of strings to be spoken when the HMI-defined 'Help' VR command is recognized (see TTSSchunk).
timeoutPrompt	Common.TTSSchunk	false	Array = true minsize = 1 maxsize = 100	This is the array of strings to be spoken when the HMI-defined timeout for VR session is about going to end (see TTSSchunk). After speaking this prompt HMI must provide some additional time for the user to be able to give a command by VR. After this additional amount of time is out, VR session must be closed.
appId	Integer	true	-	ID of the application that requested this RPC.

10.8.3 Response

Note:

There is a difference in message type for WebSocket and D-Bus connection (see the table below).

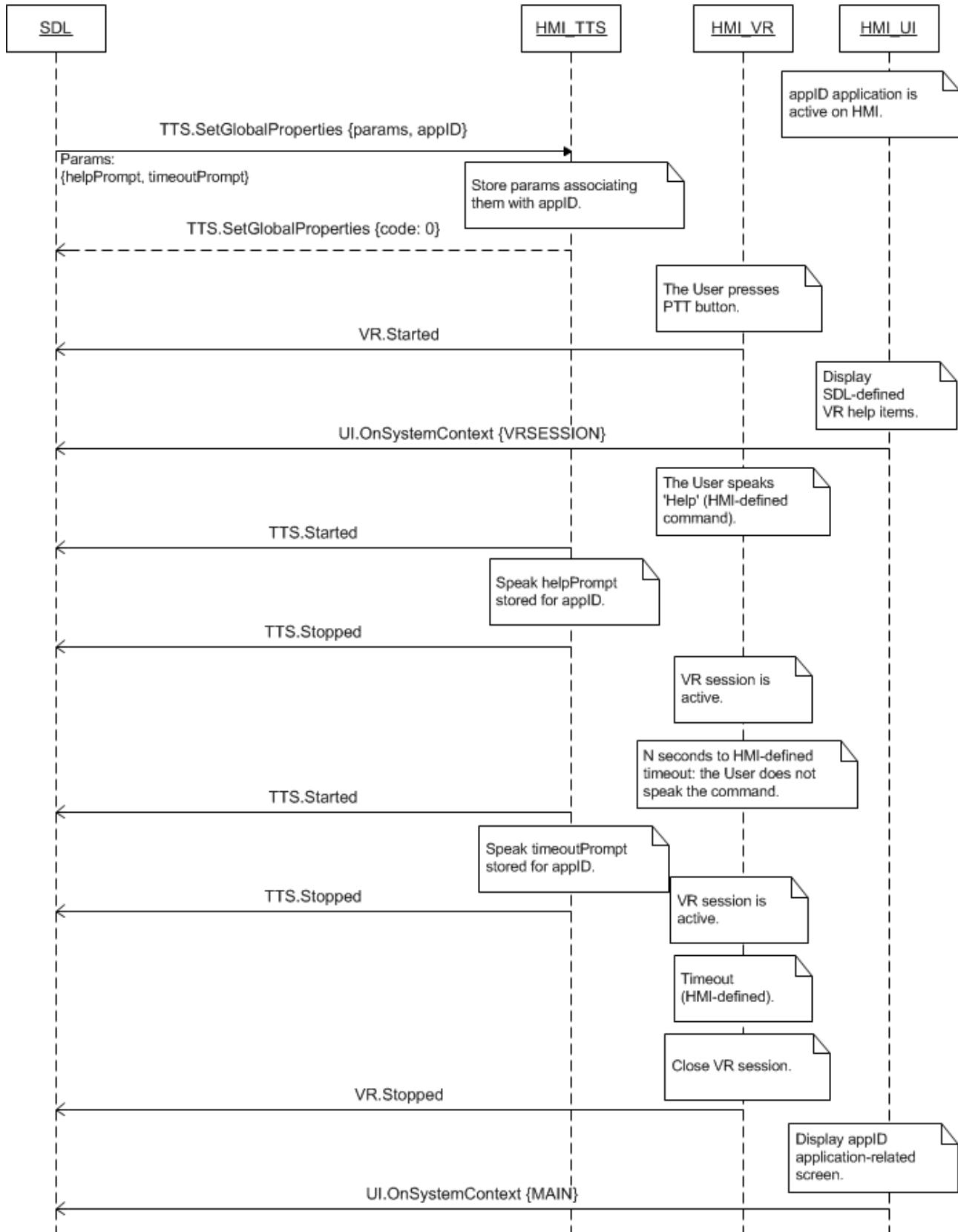
Result	Description	Message type		Message Params	Notes
		WebSocket	D-Bus		
Success	SUCCESS: The requested properties are set for TTS for the named application.	JSON response	Method return	code: 0	
Failure	UNSUPPORTED_RESOURCE The request comes for TTS which is not there on HU or doesn't support the type of phoneme provided.	JSON error message	Method return	code: 2	Applicable for this RPC result codes. Please see Result Enumeration for all SDL-supported codes.
	INVALID_ID Wrong appId sent			code: 13	
	INVALID_DATA: The data sent is invalid (invalid JSON syntax, parameters are out of bounds or of wrong type)			code: 11	
	GENERIC_ERROR 1) The unknown issue occurred or other codes are not applicable.			code: 22	

SDL 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.

10.8.4 Sequence Diagrams

10.8.4.1 TTS.SetGlobalProperties and corresponding HMI processing



Note:

SDL-defined VR help items are provided by SDL via `UI.SetGlobalProperties` RPC.

10.8.5 JSON Messages Examples

10.8.5.1 Request

```
{  
    "id" : 37,  
    "jsonrpc" : "2.0",  
    "method" : "TTS.SetGlobalProperties",  
    "params" :  
    {  
        "helpPrompt" :  
        [  
            {"text" : "Yes"},  
            {"text" : "No"},  
            {"text" : "Skip"}  
        ],  
  
        "timeoutPrompt" :  
        [  
            {"text" : "Please make a choice"},  
            {"text" : "The time is about to expire"}  
        ],  
        "appID" : 65542  
    }  
}
```

10.8.5.2 Response

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

10.8.5.3 Error message

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

10.10 OnLanguageChange

10.10.1 Description

Type:	Notification
Sender:	HMI

Purpose:	Inform about TTS language change
-----------------	----------------------------------

SDL needs to be in the know when the User changes the language of TTS: Upon the receipt of OnLanguageChange notification SDL will unregister the applications of different language to provide them with possibility to re-register with the correct (new HMI) TTS language.

HMI must:

- 1) Send the TTS.OnLanguageChange notification when the User switches TTS to another language and provide this new value via language parameter.

Note: SDL ignores all invalid notifications which come from HMI (invalid JSON, invalid data types/bounds etc)

10.10.1.1 Parameters

Param Name	Type	Mandatory	Description
language	Common.Language	true	Language TTS has been switched to. See Language

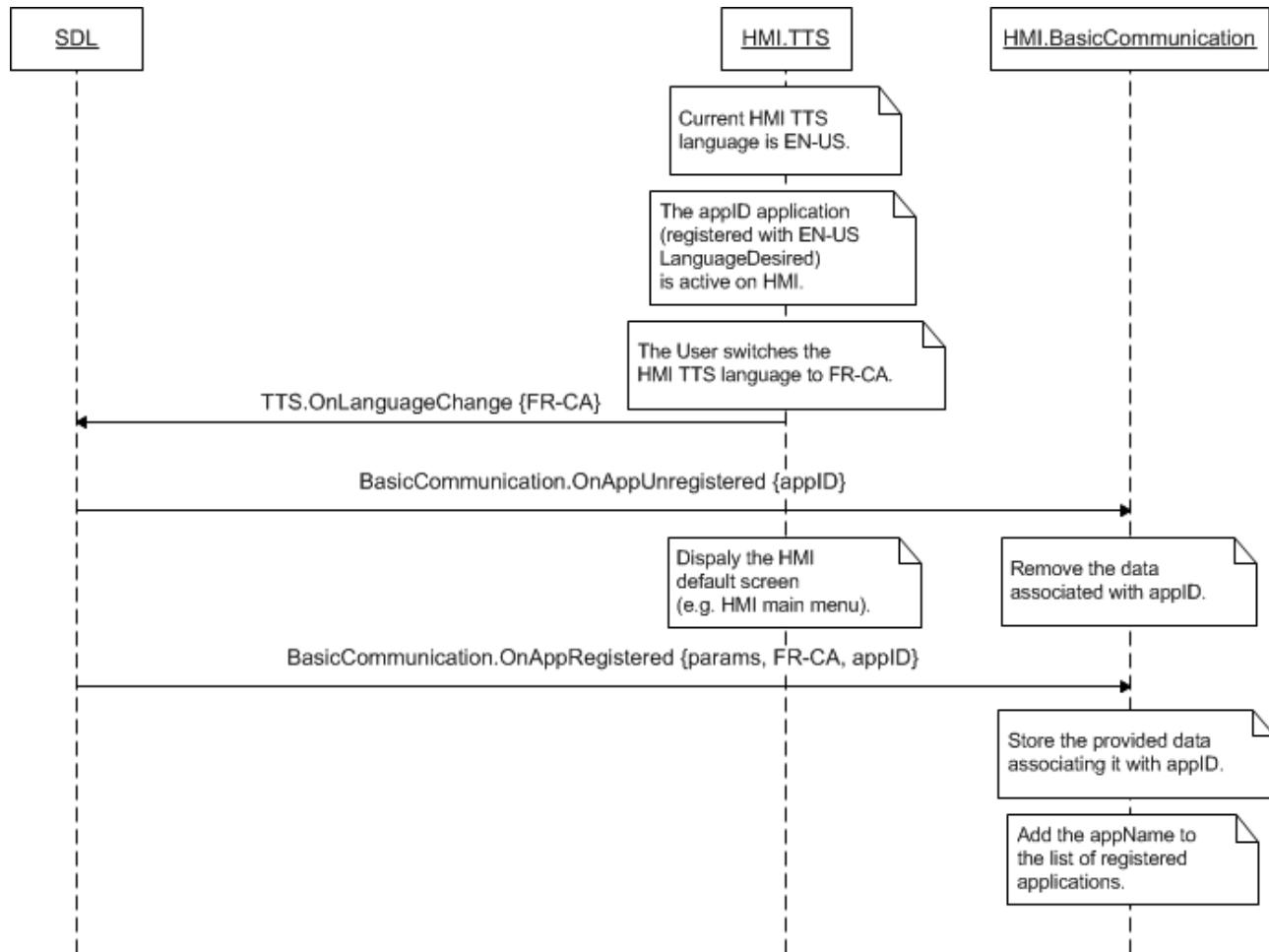
10.10.1.2 Language

Element 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

Element Name	Value	Description
JA-JP	17	Japanese – Japan
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 Belgium (Flemish)
EL-GR	25	Greek
HU-HU	26	Hungarian
FI-FI	27	Finnish
SK-SK	28	Slovak

10.10.2 Sequence Diagrams

10.10.2.1 TTS.OnLanguageChange



10.10.3 JSON Messages Examples

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

10.11 Started

10.11.1 Description

Type:	Notification
Sender:	HMI
Purpose:	Inform about TTS started

Once TTS starts speaking, HMI should attenuate the audio or make it not audible (depending on its capabilities). SDL needs to be notified about the event in order to provide a mobile application with the accurate information about audio streaming state on HMI.

HMI must:

- 1) Send TTS.Started notification when TTS has started speaking either upon SDL's request (e.g. TTS.Speak) or upon HMI-defined prompting.

Note: SDL ignores all invalid notifications which come from HMI (invalid JSON, invalid data types/bounds etc)

10.11.2 Request

10.11.2.1 Parameters

10.11.3 Response

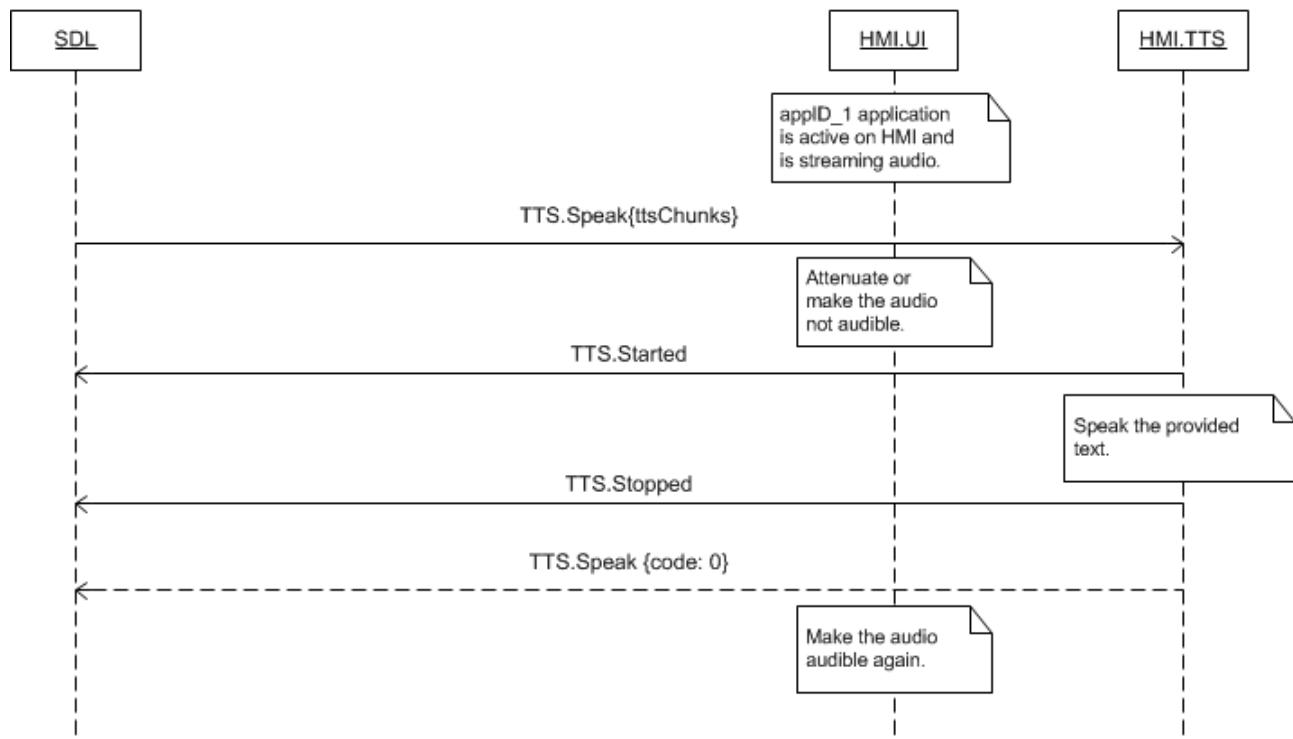
Note:

There is a difference in message type for WebSocket and D-Bus connection (see the table below).

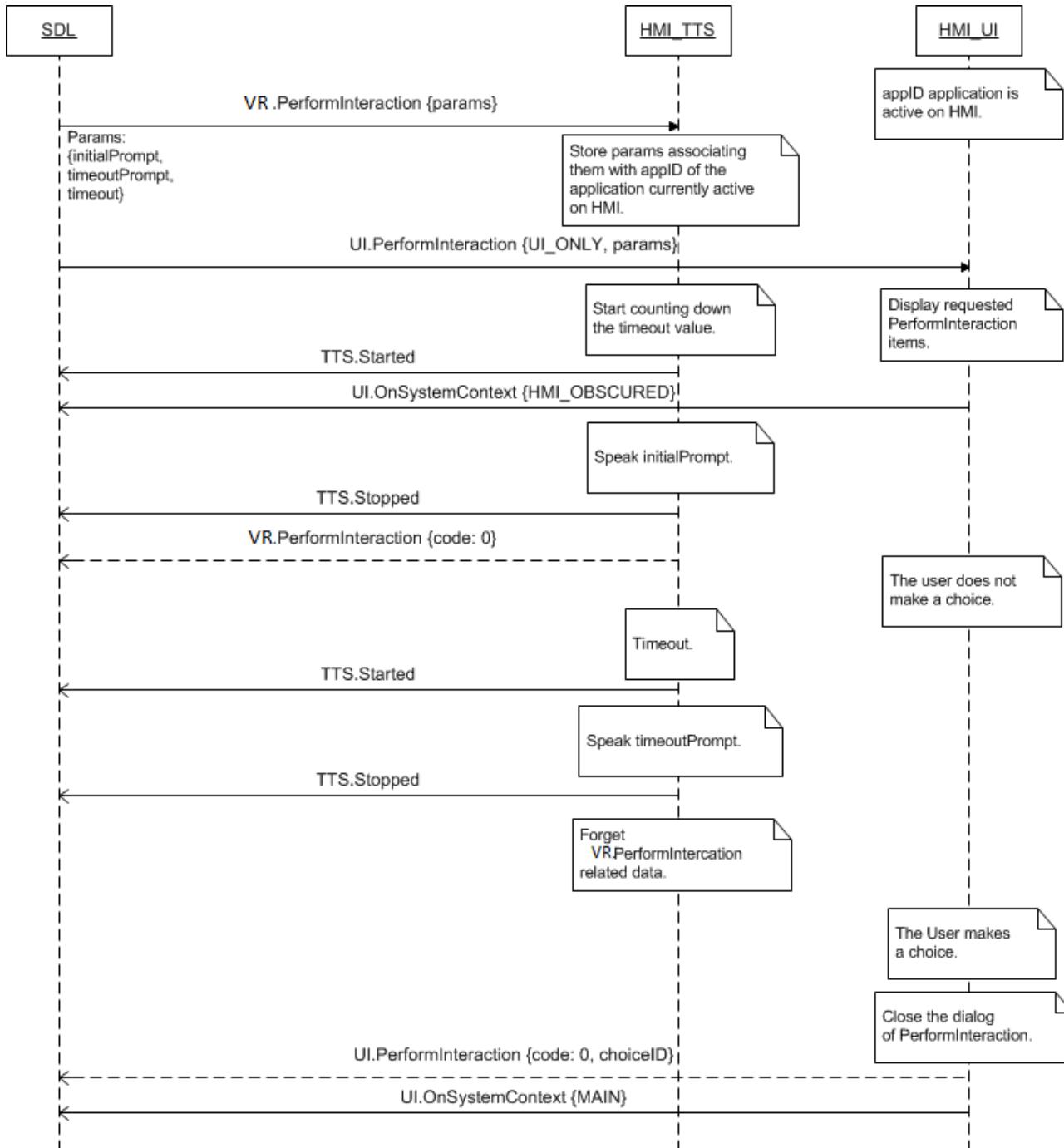
Result	Description	Message type		Message Params	Notes
		WebSocket	D-Bus		
Success	SUCCESS: The requested properties are set for TTS for the named application.	JSON response	Method return	code: 0	
Failure	GENERIC_ERROR: The unknown issue occurred or other codes are not applicable.	JSON error message	Method return	code: 22	

10.11.3 Sequence Diagrams

10.11.3.1 TTS.Started upon TTS.Speak request from SDL



10.11.3.2 TTS.Started during PerformInteraction



10.11.4 JSON Messages Examples

10.11.4.1 Request

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

10.11.4.2 Response

```
{
  "id" : 37,
  "jsonrpc" : "2.0",
```

```

    "result" :
    {
        "code" : 0,
        "method" : "TTS.Started"
    }
}

```

10.11.4.3 Error message

```

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

```

10.12 Stopped

10.12.1 Description

Type:	Notification
Sender:	HMI
Purpose:	Inform about TTS has stopped speaking

When TTS module has finished speaking (started whatever by SDL via e.g. TTS.Speak or by HMI itself), HMI must notify SDL about the event via TTS.Stopped.

HMI must:

Send TTS.Stopped notification when TTS has stopped speaking

- On The designated text
- Having been aborted by
 - SDL (e.g. TTS.StopSpeaking)
 - User (e.g. PTT button press)
 - Event of higher priority (e.g. system or SDL-defined alert message)

Note: SDL ignores all invalid notifications which come from HMI (invalid JSON, invalid data types/bounds etc)

10.12.2 Request

10.12.2.1 Parameters

10.12.3 Response

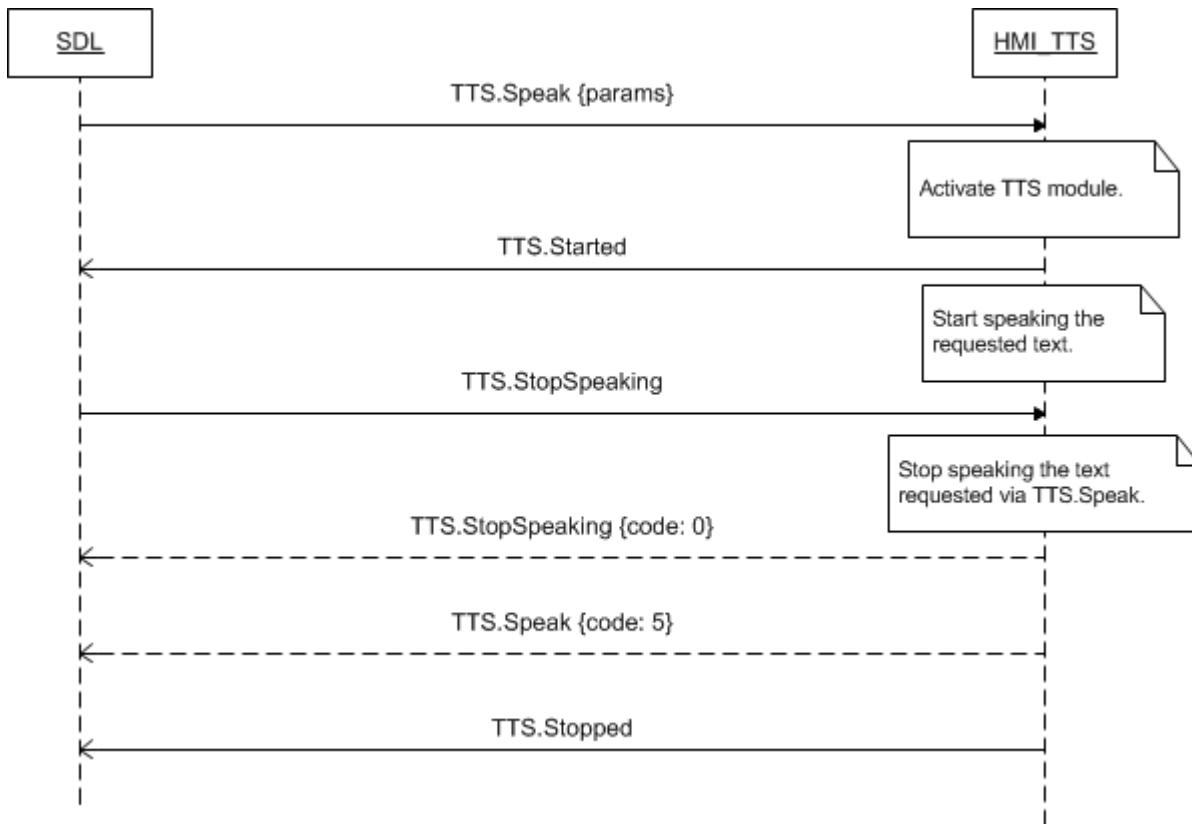
Note:

There is a difference in message type for WebSocket and D-Bus connection (see the table below).

Result	Description	Message type		Message Params	Notes
		WebSocket	D-Bus		
Success	SUCCESS: The requested properties are set for TTS for the named application.	JSON response	Method return	code: 0	
Failure	GENERIC_ERROR: The unknown issue occurred or other codes are not applicable.	JSON error message	Method return	code: 22	

10.12.4 Sequence Diagrams

10.12.4.1 TTS.Stopped after TTS.StopSpeaking from SDL



10.12.5 JSON Messages Examples

10.12.5.1 Request

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

10.12.5.2 Response

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

10.12.5.3 Error message

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

10.13 OnResetTimeout

10.13.1 Description

Type:	Notification
Sender:	HMI
Purpose:	Inform about timeout reset on TTS for defined running RPC

SDL's default timeout for any response from HMI is 10 seconds. So it needs to know if some TTS RPC is still running on HMI to predict sending wrong response by timeout to mobile application. OnResetTimeout notifies SDL to reset the timeout for some TTS RPC running on HMI (e.g.Speak).

HMI must:

Send the OnResetTimeout notification in the following cases:

- In case HMI speak event is going longer than 10 seconds

Note: SDL ignores all invalid notifications which come from HMI (invalid JSON, invalid data types/bounds etc)

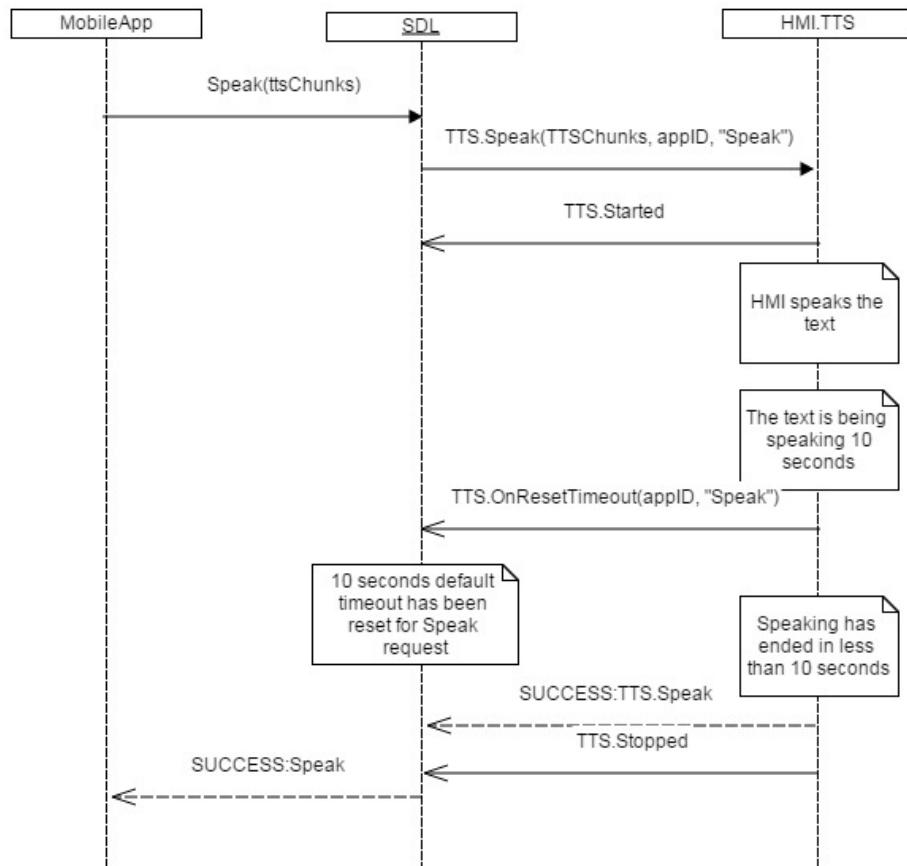
10.13.1.1 Parameters

Param Name	Type	Mandatory	Description
appID	Integer	true	Id of application that invoked notification.

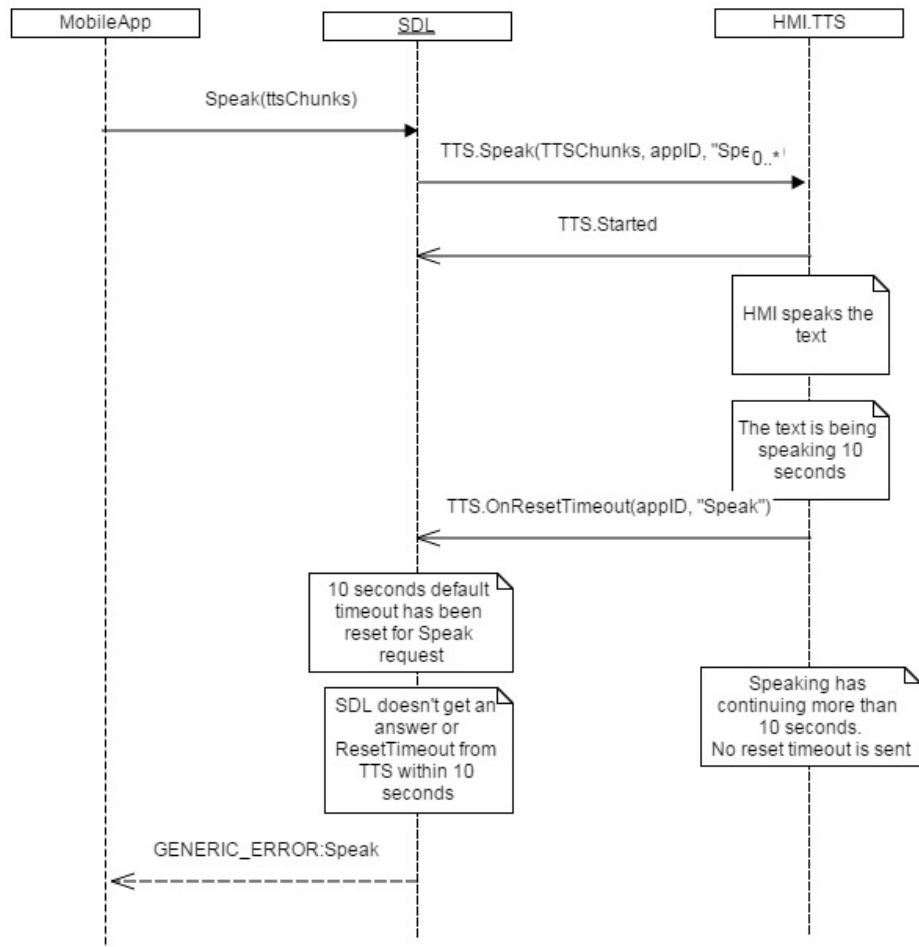
methodName	String	true	Method name which is still active on TTS and requires timeout reset on SDL
------------	--------	------	--

10.13.2 Sequence Diagrams

10.13.2.1 OnResetTimeout for TTS.Speak SUCCESS



10.13.2.2 OnResetTimeout for TTS.Speak GENERIC ERROR



10.13.3 JSON Messages Examples

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

11 VehicleInfo Component Description

11.1 IsReady

11.1.1 Description

Type:	Function
Sender:	SDL
Purpose:	Get information if Vehicle component is ready to communicate

The request comes after HMI's readiness is confirmed via [OnReady](#) notification. SDL requires the information about whether the vehicle information can be collected and provided by HMI.

Note:

If VehicleInfo component is responded to be unavailable, SDL will not further send the requests related to it.

11.1.2 Request

11.1.2.1 Behavior

HMI must:

- 1) Check whether VehicleInfo component is present and ready to communicate with SDL
- 2) Respond correspondingly to results of this check.

11.1.3 Response

Note:

There is a difference in message type for WebSocket and D-Bus connection (see the table below).

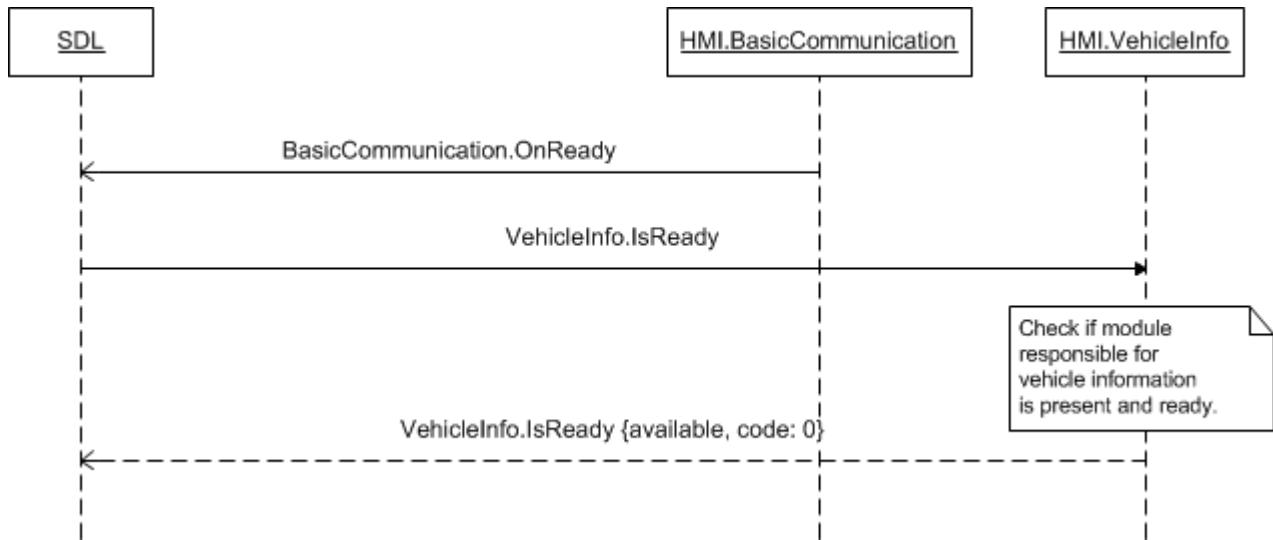
Result	Description	Message type		Message Params	Notes
		WebSocket	D-Bus		
Success	SUCCESS: HMI provides the information about VehicleInfo availability.	JSON response	Method return	available, code: 0	
Failure	INVALID_DATA: The data sent is invalid (invalid JSON syntax)	JSON error message	Method return	code: 11	Applicable for this RPC result codes.
	DATA_NOT_AVAILABLE: The information about VehicleInfo availability cannot be provided.			Code: 9	
	GENERIC_ERROR: The unknown issue occurred or other codes are not applicable.			code: 22	Please see Result Enumeration for all SDL-supported codes.

11.1.3.1 Parameters

Param Name	Type	Mandatory	Description
availabe	Boolean	true	Must be - 'true' if VehicleInfo component is present and ready to operate - 'false' if not.

11.1.4 Sequence Diagrams

11.1.4.1 VehicleInfo.IsReady and preceding OnReady



11.1.5 JSON Messages Examples

11.1.5.1 Request

```
{
  "id" : 17,
  "jsonrpc" : "2.0",
  "method" : "VehicleInfo.IsReady"
}
```

11.1.5.2 Response

```
{
  "id" : 17,
  "jsonrpc" : "2.0",
  "result" :
  {
    "available" : true,
    "code" : 0,
    "method" : "VehicleInfo.IsReady"
  }
}
```

11.1.5.3 Error message

```
{
  "id" : 17,
  "jsonrpc" : "2.0",
  "error" :
  {
    "code" : 9,
    "message" : "Data not available",
    "data" :
    {
    }
  }
}
```

```

        "method" :
    "VehicleInfo.IsReady"
    }
}
}

```

11.2 GetVehicleType

11.2.1 Description

Type:	Function
Sender:	SDL
Purpose:	Get general information about vehicle.

SDL can store the information about the type of the host vehicle in `smartDeviceLink.ini` file stored in SDL directory (see `HMICapabilities` parameter of ini file). Still in current implementation, SDL will request the information via `VehicleInfo.GetVehicleType` after getting `OnReady` notification and response to `VehicleInfo.IsReady` from HMI.

SDL Note: In case SDL didn't get VehicleType from HMI, it uses the default one from the file, defined in `smartDeviceLink.ini` as `HMICapabilities` parameter.

11.2.2 Request

11.2.2.1 Behavior

HMI must:

- 1) Check for the information requested (make, model, trim and model year).
- 2) Provide this information in response message (see section 11.2.3).

11.2.3 Response

Note:

There is a difference in message type for WebSocket and D-Bus connection (see the table below).

Result	Description	Message type		Message Params	Notes
		WebSocket	D-Bus		
Success	SUCCESS: HMI provides the information about vehicle.	JSON response	Method return	vehicleType, code: 0	
Failure	INVALID_DATA: The data sent is invalid (invalid JSON syntax)	JSON error message	Method return	code: 11	Applicable for this RPC result codes.
	DATA_NOT_AVAILABLE: The information about vehicle cannot be provided.			Code: 9	Please see Result Enumeration for all SDL-

	GENERIC_ERROR: The unknown issue occurred or other codes are not applicable.			code: 22	supported codes.
--	---	--	--	----------	------------------

11.2.3.1 Parameters

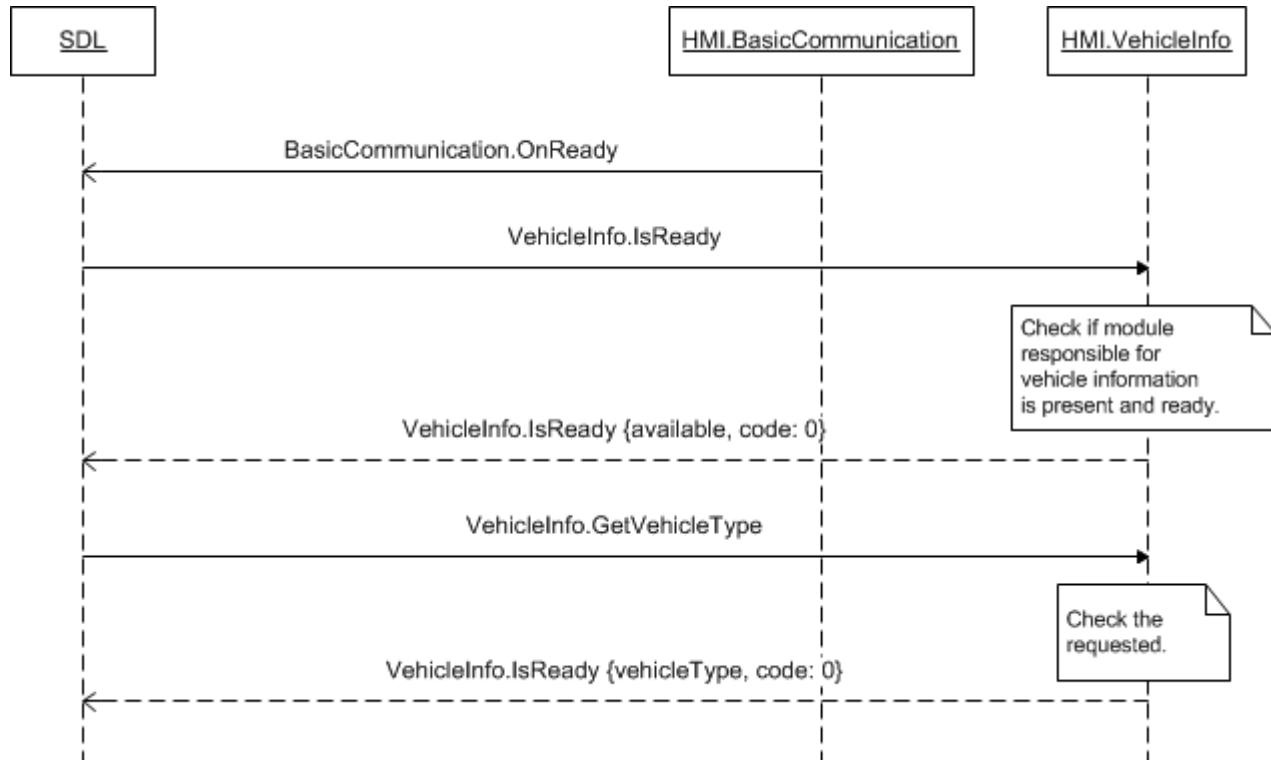
Param Name	Type	Mandatory	Description
vehicleType	Common.VehicleType	true	See VehicleType

11.2.3.2 VehicleType

Param Name	Type	Mandatory	Additional	Description
make	String	false	maxlength = 500	Make of the vehicle (e.g. Ford)
model	String	false	maxlength = 500	Model of the vehicle (e.g. Fiesta)
modelYear	String	false	maxlength = 500	Model Year of the vehicle (e.g. 2013)
trim	String	false	maxlength = 500	Trim of the vehicle (e.g. SE)

11.2.4 Sequence Diagrams

11.2.4.1 GetVehicleType



11.2.5 JSON Messages Examples

11.2.5.1 Request

```
{  
    "id" : 21,  
    "jsonrpc" : "2.0",  
    "method" :  
    "VehicleInfo.GetVehicleType"  
}
```

11.2.5.2 Response

```
{  
    "id" : 21,  
    "jsonrpc" : "2.0",  
    "result" :  
    {  
        "vehicleType" :  
        [  
            {"make" : "Ford",  
             "model" : "Fusion",  
             "modelYear" : "2013",  
             "trim" : "SE"}  
        ]  
    }  
    "code" : 0,  
    "method" :  
    "VehicleInfo.GetVehicleType"  
}
```

11.2.5.3 Error message

```
{  
    "id" : 21,  
    "jsonrpc" : "2.0",  
    "error" :  
    {  
        "code" : 9,  
        "message" : "The requested data is  
not available",  
        "data" :  
        {  
            "method" :  
            "VehicleInfo.GetVehicleType"}  
    }  
}
```

11.3 ReadDID

11.3.1 Description

Type:	Function
Sender:	SDL
Purpose:	Receive the known DIDs.

SDL requests to provide the known DIDs, i.e. Data Identifiers: addressed memory locations of the Electronic Control Unit (ECU).

11.3.2 Request

11.3.2.1 Behavior

HMI must:

- 1) Check data in the requested locations (from `didLocation` parameter, which may be the array of addresses) on the named ECU (`ecuName` parameter).
- 2) Respond with one of the appropriate result codes (see section 11.3.3 Response for applicable result codes). And in case of SUCCESS return the array of `didResult`, each element of which refers the data from one of the requested locations:
 - Data itself (`data` parameter in `DIDResult` structure), which is the hex byte string of however many bytes stored at named location. May be missing in case of any kind of error occurred (see section 11.3.3.3 `VehicleDataresultCode`).
 - Address where it is taken from (`didLocation` parameter). Must be equal to the value from request.
 - Individual result code (`resultCode` parameter), which defines the status of the data being provided.

SDL Note: In case HMI does not respond SDL's request during SDL default timeout (10 sec), SDL will return the `GENERIC_ERROR` result code to the corresponding mobile app's request

11.3.2.1 Parameters

Param Name	Type	Mandatory	Additional	Description
<code>ecuName</code>	Integer	true	<code>Minvalue = 0</code> <code>Maxvalue = 65535</code>	Name of ECU.
<code>didLocation</code>	Integer	true	<code>Array = true</code> <code>Minsize = 1</code> <code>Maxsize = 1000</code> <code>Minvalue = 0</code> <code>Maxvalue = 65535</code>	The array of addresses of DID location on ECU.
<code>appID</code>	Integer	true	—	ID of the application that requested this RPC.

11.3.3 Response

Note:

There is a difference in message type for WebSocket and D-Bus connection (see the table below).

Result	Description	Message type		Message Params	Notes
		WebSocket	D-Bus		

Success	SUCCESS: HMI provides requested DIDs.	JSON response	Method return	didResult, code: 0	
Failure	UNSUPPORTED_RESOURCE: The named ECU does not exist.	JSON error message	Method return	code: 2	Applicable for this RPC result codes. Please see Result Enumeration for all SDL-supported codes.
	TOO_MANY_PENDING_REQUESTS Diagnostic module is overflow or busy			code: 18	
	REJECTED: 1. The named ECU exists, but <u>all</u> of requested DIDs data is unavailable. 2. The HU System is busy with a higher priority event and rejects this RPC.			code: 4	
	INVALID_DATA: The data sent is invalid (invalid JSON syntax, parameters out of bounds or of wrong type)			code: 11	
	DATA_NOT_AVAILABLE Some of the data requested is not available (but not <u>all</u>)			code: 9	
	INVALID_ID Wrong appID sent			code: 13	
	GENERIC_ERROR 1) The unknown issue occurred or other codes are not applicable.			code: 22	
	TRUNCATED_DATA: One or more individual resultCodes (see DIDResult struct) contains TRUNCATED_DATA.			didResult, Code: 24	

11.3.3.1 Parameters

Param Name	Type	Mandatory	Additional	Description
didResult	Common.DIDResult	false	Array = true Minsize = 0 Maxsize = 1000	Array of requested DID results (with data if available). See DIDResult

11.3.3.2 DIDResult

Param Name	Type	Mandatory	Additional	Description
------------	------	-----------	------------	-------------

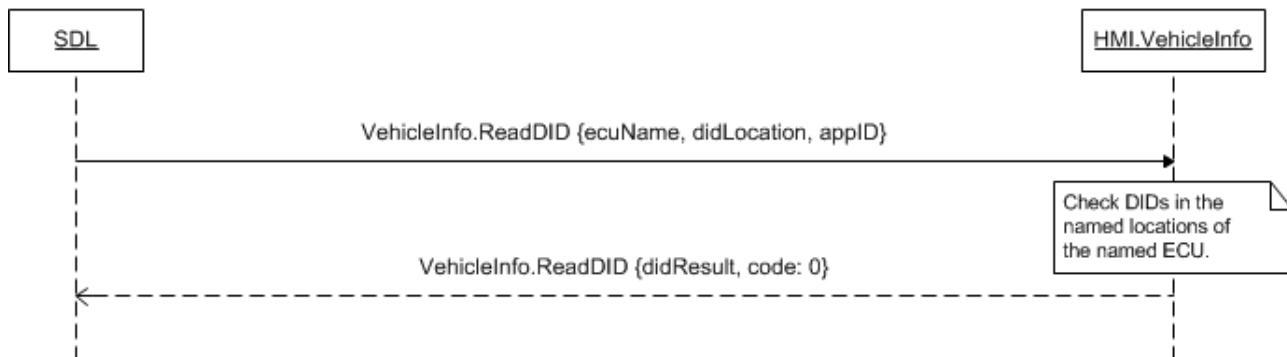
Param Name	Type	Mandatory	Additional	Description
resultCode	Common.VehicleDataResultCode	true	-	Individual DID result code. See VehicleDataResultCode.
didLocation	Integer	true	minvalue = 0 maxvalue = 65535	The address of DID location from the corresponding request.
data	String	false	maxlength = 5000	The DID data which is the hex byte string of however many bytes are stored at that location

11.3.3.3 VehicleDataResultCode

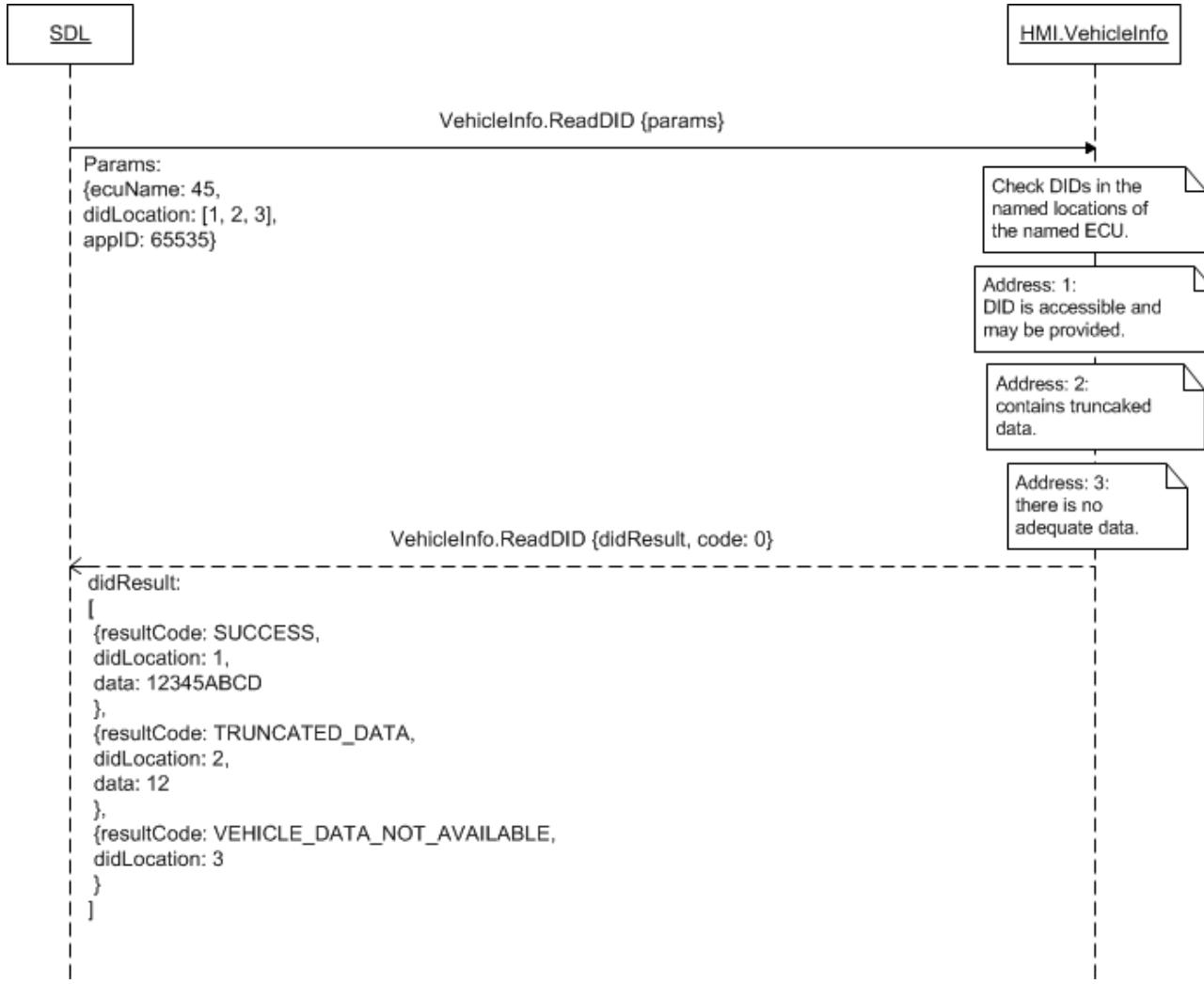
Element name	Value	Short Description
SUCCESS	0	The requested data is accessible.
TRUNCATED_DATA	1	The data is truncated: not all of the requested information is available.
DISALLOWED	2	Not applicable
USER_DISALLOWED	3	The request is included in a functional group explicitly blocked by the User.
INVALID_ID	4	One of the provided IDs is not valid.
VEHICLE_DATA_NOT_AVAILABLE	5	The requested data is not available.
DATA_ALREADY_SUBSCRIBED	6	Not applicable
DATA_NOT_SUBSCRIBED	7	Not applicable
IGNORED	8	Not applicable

11.3.4 Sequence Diagrams

11.3.4.1 ReadDID general processing



11.3.4.2 ReadDID with expanded didResult in response



11.3.5 JSON Messages Examples

11.3.5.1 Request

```
{
  "id" : 158,
  "jsonrpc" : "2.0",
  "method" : "VehicleInfo.ReadDID",
  "params" :
  {
    "ecuName" : 1287,
    "didLocation" : [35, 48, 182],
    "appId" : 93
  }
}
```

11.3.5.2 Response

```
{
  "id" : 158,
  "jsonrpc" : "2.0",
  "result" :
  {
    "didResult" :
    [
      {
        "did": 1287,
        "data": "12345ABCD"
      }
    ]
  }
}
```

```

        "resultCode" : SUCCESS,
        "didLocation" : 35,
        "data" : "38AF"
    },
    {
        "resultCode" :
TRUNCATED_DATA,
        "didLocation" : 48,
        "data" : "35"
    },
    {
        "resultCode" : INVALID_ID,
        "didLocation" : 182
    }
],
"code" : 0,
"method" : "VehicleInfo.ReadDID"
}
}

```

11.3.5.3 Error message

```
{
    "id" : 158,
    "jsonrpc" : "2.0",
    "error" :
    {
        "code" : 2,
        "message" : "The requested ECU does
not exist",
        "data" :
        {
            "method" :
"VehicleInfo.ReadDID"
        }
    }
}
```

11.4 GetDTCs

11.4.1 Description

Type:	Function
Sender:	SDL
Purpose:	Receive the known DTCs.

SDL requests to provide the known Diagnostic Trouble Codes (DTCs) of the Electronic Control Unit (ECU).

11.4.2 Request

11.4.2.1 Behavior

HMI must:

1. Check the requested DTCs using provided information of `ecuName` and `dtcMask`.
2. Respond with one of the appropriate result codes (see section 11.4.3 Response for applicable result codes). And in case of `SUCCESS` return the following:

- 2 byte ECU Header (`ecuHeader` parameter) that must contain the information whether the below list of DTCs is truncated or not.
- Array of all reported DTCs on module (`dtc` parameter), each of the elements is expected to consist of 4 bytes: 3 bytes of data and 1 byte of status.

SDL 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

11.4.2.1 Parameters

Param Name	Type	Mandatory	Additional	Description
ecuName	Integer	true	minvalue = 0 maxvalue = 65535	Name of ECU.
dtcMask	Integer	false	minvalue = 0 maxvalue = 255	DTC Mask Byte to be sent in diagnostic request to module
appID	Integer	true	-	ID of the application that requested this RPC.

11.4.3 Response

Note:

There is a difference in message type for WebSocket and D-Bus connection (see the table below).

Result	Description	Message type		Message Params	Notes
		WebSocket	D-Bus		
Success	SUCCESS: HMI provides requested DTCs.	JSON response	Method return	ecuHeader, dtc, code: 0	
Failure	TOO_MANY_PENDING_REQUESTS Diagnostic module is overflow or busy	JSON error message	Method return	code: 18	Applicable for this RPC result codes. Please see Result Enumeration for all SDL-supported codes.
	UNSUPPORTED_RESOURCE: The named ECU does not exist.			code: 2	
	REJECTED: 1. The named ECU exists, but <u>all</u> of requested DIDs data is unavailable. 2. The HU System is busy with a higher priority event and rejects this RPC.			code: 4	
	DATA_NOT_AVAILABLE Some of the data requested is not available (but not <u>all</u>)			code: 9	
	INVALID_ID Wrong appID sent			code: 13	

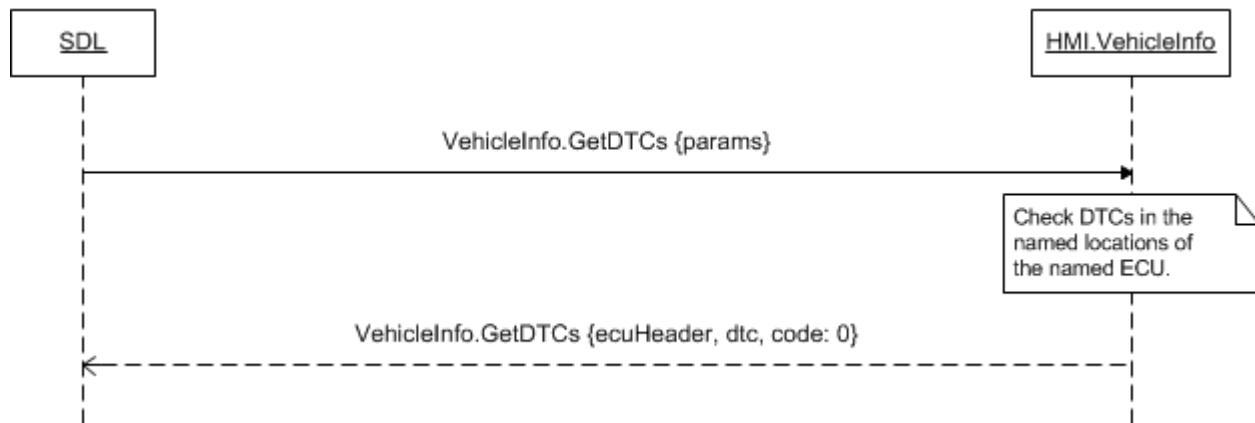
	1) The unknown issue occurred or other codes are not applicable.		code: 22	
	TRUNCATED_DATA: One or more DTCs in the array is truncated .		ecuHeader, dtc, Code: 24	

11.4.3.1 Parameters

Param Name	Type	Mandatory	Additional	Description
ecuHeader	Integer	true	minvalue = 0 maxvalue = 65535	2 byte ECU Header for DTC response. Contains the information whether the below list of DTCs is truncated.
dtc	String	false	Array = true Minsize = 1 Maxsize = 15 Maxlength = 10	Array of all reported DTCs on module. Each DTC is represented with 4 bytes: - 3 bytes for data - 1 byte for status

11.4.4 Sequence Diagrams

11.4.4.1 GetDTCs



11.4.5 JSON Messages Examples

11.4.5.1 Request

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

11.4.5.2 Response

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

11.4.5.3 Error message

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

11.5 DiagnosticMessage

11.5.1 Description

Type:	Function
Sender:	SDL
Purpose:	Vehicle diagnostic

Non periodic vehicle diagnostic request.

Note: Allowed to be requested *messageData* values are listed in [smartDeviceLink.ini file in \[MAIN\] section](#), parameter SupportedDiagModes. The request came from mobile application must be validated by SDL.

In case the request contains the values non-listed in smartDeviceLink.ini file, it will be REJECTED by SDL, otherwise the requested data must be transferred to HMI via DiagnosticMessage API.

Example:

SupportedDiagModes = 0x01, 0x02, 0x03, 0x05, 0x06, 0x07, 0x09, 0x0A, 0x18, 0x19, 0x22, 0x3E

11.5.2 Request

11.5.2.1 Behavior

HMI must:

- 1) Check the requested data using provided information of targetID (name of ECU), messageLength and messageData.
- 2) Respond with one of the appropriate result codes (see section 11.5.3 Response for applicable result codes). And in case of SUCCESS return messageDataResult which is an array of bytes comprising CAN message result.

SDL 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.

11.5.2.1 Parameters

Param Name	Type	Mandatory	Additional	Description
targetID	Integer	true	Minvalue = 0 Maxvalue = 65535	Name of target ECU.
messageLength	Integer	true	Minvalue = 0 Maxvalue = 65535	Length of message (in bytes).
messageData	Integer	true	Array = true Minsize = 1 Maxsize = 65535 Minvalue = 0 Maxvalue = 255	Array of bytes comprising CAN message.
appID	Integer	true	-	ID of application that requested this RPC.

11.5.3 Response

Note:

There is a difference in message type for WebSocket and D-Bus connection (see the table below).

Result	Description	Message type		Message Params	Notes
		WebSocket	D-Bus		
Success	SUCCESS: HMI provides requested DTCs.	JSON response	Method return	messageDataResult, code: 0	
Failure	UNsupported RESOURCE: The named ECU does not exist.	JSON error message	Method return	code: 2	Applicable for this RPC result codes.
	TOO_MANY_PENDING_REQUESTS Diagnostic module is overflow or busy			code: 18	
	REJECTED: The HU System is busy with a higher priority event and rejects this RPC.			Code: 4	Please see Result Enumeration for all SDL-supported codes.
	DATA_NOT_AVAILABLE The data requested is not available			code: 9	

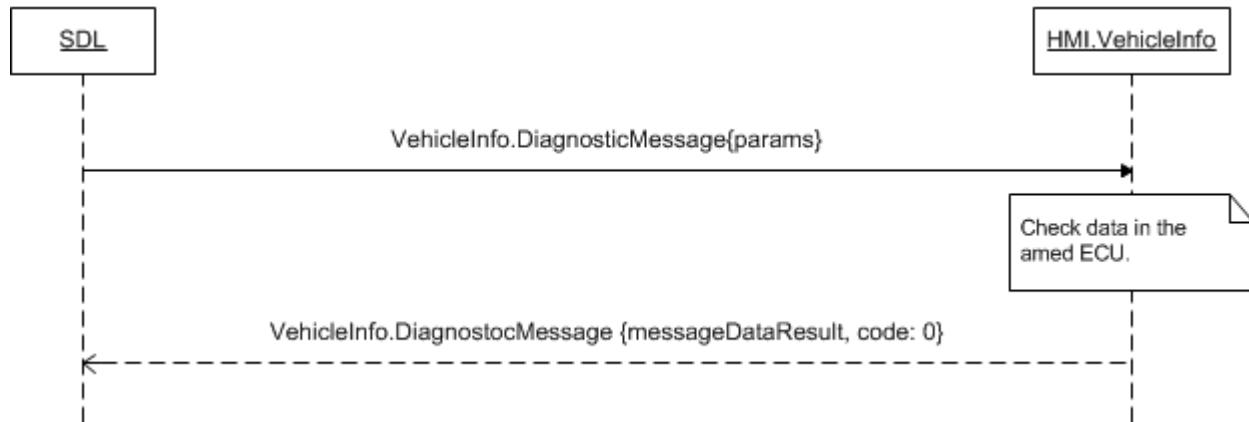
	INVALID_ID Wrong appID sent			code:13	
	GENERIC_ERROR 1) The unknown issue occurred or other codes are not applicable.			code: 22	
	TRUNCATED_DATA: One or more messages in the array is truncated			messageDataResult, code: 24	

11.5.3.1 Parameters

Param Name	Type	Mandatory	Additional	Description
messageDataResult	Integer	true	Array = true Minsize = 1 Maxsize = 65535 Minvalue = 0 Maxvalue = 255	Array of bytes comprising CAN message result.

11.5.4 Sequence Diagrams

11.5.4.1 DiagnosticMessage



11.5.5 JSON Messages Examples

11.5.5.1 Request

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

```
}
```

11.5.5.2 Response

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

11.5.5.3 Error message

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

11.6 SubscribeVehicleData

11.6.1 Description

Type:	Function
Sender:	SDL
Purpose:	Subscribe for definite vehicle data types updates

SDL provides the data types on which periodic updates an application would like to subscribe. In case of no errors, SDL expects to receive the information via `OnVehicleData` from HMI whenever the subscribed data type gets updates.

The request may come for the application being whatever active or in background on HMI (regulated by SDL Policy Manager).

11.6.2 Request

11.6.2.1 Behavior

HMI must:

1. Check whether the periodic updates can be provided for requested data type. The data type is defined with Boolean parameters of `gps`, `speed`, `rpm` and other that are provided in section 11.6.2.2:

- Parameters of `true` value have to be subscribed
- One or several or all of the parameters from section 11.6.2.2 may be included into request..

Note:

Within a request SDL will provide at least one ‘ture’-value parameter of data type to be subscribed.

2. Remember the requested data type in the subscription list and notify SDL via `OnVehicleData` in case of subscribed data values changes (see 11.6.2.2).

3. Respond with one of appropriate result codes (see section [11.6.3 Response](#)). And in case of `SUCCESS` return the parameter(s) of the name(s) equal to those requested for subscription. The returned parameters must contain

- Data type (see [section 11.6.3.3](#)) correspondent to the name of related parameter (for example, `gps : {VEHICLEDATA_GPS, SUCCESS}`, `speed : {VEHICLEDATA_SPEED, SUCCESS}`).
- And individual `VehicleDataResult` code from [section 11.6.3.4](#). The codes different from `SUCCESS` will inform SDL about the status of the named data type and whether the updates would be provided.

4. Provide `OnVehicleData` notifications whenever the successfully subscribed vehicle data type provide data changes until SDL unsubscribes from updates via `UnsubscribeVehicleData` (see section 11.7).

SDL Note: HMI is not being provided with the information which one application requests the data subscription, SDL manages this internally and redistributes data received via `OnVehicleData` between the subscribed applications by itself.

In case HMI does not respond SDL's request during SDL default timeout (10 sec), SDL will return the result `GENERIC_ERROR` to the corresponding mobile app's request

11.6.2.2 Parameters

Param Name	Type	Mandatory	Description
<code>gps</code>	Boolean	false	Subscribe for GPS data updates. Information related to GPS data: number of satellites, compass direction, longitude, latitude, etc. See <code>GPSData</code>
<code>speed</code>	Boolean	false	Subscribe for vehicle speed updates to be provided in kilometers per hour.
<code>rpm</code>	Boolean	false	Subscribe for the number of engine revolutions per minute updates.
<code>fuelLevel</code>	Boolean	false	Subscribe for the fuel level in the tank updates (percentage).
<code>fuelLevel_State</code>	Boolean	false	Subscribe for the fuel level state updates: normal, low, etc. See <code>ComponentVolumeStatus</code> .
<code>instantFuelConsumption</code>	Boolean	false	Subscribe for the instantaneous fuel consumption updates to be provided in microliters.

Param Name	Type	Mandatory	Description
externalTemperature	Boolean	false	Subscribe for the external temperature to be provided in degrees Celsius.
prndl	Boolean	false	Subscribe for gear stick position updates: first, second, etc. See PRNDL.
tirePressure	Boolean	false	Subscribe for tire pressure status updates: - The warning status (on, off, etc.) - The status of the tire pressure itself: normal, low, etc. See TireStatus.
odometer	Boolean	false	Subscribe for the information from odometer to be provided in km.
beltStatus	Boolean	false	Subscribe for the information of the seat belts status updates: deployed, buckled. See BeltStatus.
bodyInformation	Boolean	false	Subscribe for updates of body information including power modes: park brake status, ignition and ignition stable status. See BodyInformation.
deviceStatus	Boolean	false	Subscribe for updates of device status including signal and battery strength. See DeviceStatus.
driverBraking	Boolean	false	Subscribe for the information of the brake pedal status updates: on, off, etc. See VehicleDataEventStatus.
wiperStatus	Boolean	false	Subscribe for updates of the wipers status: when they are manually on, manually off, stalled, etc. See WiperStatus.
headLampStatus	Boolean	false	Subscribe for the updates of the head lamps status: when they are on, off, etc. See HeadLampStatus.
engineTorque	Boolean	false	Subscribe for the updates of the torque value for engine (in Nm) on non-diesel variants
accPedalPosition	Boolean	false	Subscribe for the information of accelerator pedal position (percentage depressed).
steeringWheelAngle	Boolean	false	Subscribe for the information of current angle of the steering wheel (in degrees)

11.6.3 Response

Note:

There is a difference in message type for WebSocket and D-Bus connection (see the table below).

Result	Description	Message type		Message Params	Notes
		WebSocket	D-Bus		
Success	SUCCESS: HMI processes the request successfully and returns the information about the data type(s) and the corresponding individual subscription result code (see p.3 of section 11.6.2.1)	JSON response	Method return	Parameters of the names equal to requested ones, code: 0	See 11.6.3.1 Parameters.
Failure	IGNORED: Subscription is requested for the single data which is already subscribed	JSON error message	Method return	Parameters of the names equal to requested ones code: 6	Applicable for this RPC result codes.

	INVALID_ID Wrong appID			code:13	Please see Result Enumeration for all SDL- supported codes.
	GENERIC_ERROR 1) The unknown issue occurred or other codes are not applicable.			code: 22	

11.6.3.1 Parameters

Param Name	Type	Mandatory	Description		
gps	Common.VehicleDataResult	false	GPS data subscription status. See section 11.6.3.2 VehicleDataResult (VEHICLEDATA_GPS must be used).		
speed	Common.VehicleDataResult	false	Vehicle speed subscription status. See section 11.6.3.2 VehicleDataResult (VEHICLEDATA_SPEED must be used).		
rpm	Common.VehicleDataResult	false	The engine number of revolutions per minute data subscription status. See section 11.6.3.2 VehicleDataResult (VEHICLEDATA_RPM must be used).		
fuelLevel	Common.VehicleDataResult	false	The fuel level data subscription status. See section 11.6.3.2 VehicleDataResult (VEHICLEDATA_FUELLEVEL must be used).		
fuelLevel_State	Common.VehicleDataResult	false	The fuel level state data subscription status. See section 11.6.3.2 VehicleDataResult (VEHICLEDATA_FUELLEVEL_STATE must be used).		
instantFuelConsumption	Common.VehicleDataResult	false	The instantaneous fuel consumption data subscription status. See section 11.6.3.2 VehicleDataResult (VEHICLEDATA_FUELCONSUMPTION must be used).		
externalTemperature	Common.VehicleDataResult	false	The external temperature data subscription status. See section 11.6.3.2 VehicleDataResult (VEHICLEDATA_EXTERNTTEMP must be used).		
prndl	Common.VehicleDataResult	false	The gear stick position data subscription status. See section 11.6.3.2 VehicleDataResult (VEHICLEDATA_PRNDL must be used).		
tirePressure	Common.VehicleDataResult	false	The tire pressure data subscription status. See section 11.6.3.2 VehicleDataResult (VEHICLEDATA_TIREPRESSURE must be used).		
odometer	Common.VehicleDataResult	false	Odometer data subscription status. See section 11.6.3.2 VehicleDataResult (VEHICLEDATA_ODOMETER must be used).		
beltStatus	Common.VehicleDataResult	false	The seat belts status data subscription status. See section 11.6.3.2 VehicleDataResult (VEHICLEDATA_BELTSTATUS must be used).		
bodyInformation	Common.VehicleDataResult	false	The body information data subscription status. See section 11.6.3.2 VehicleDataResult (VEHICLEDATA_BODYINFO must be used).		
deviceStatus	Common.VehicleDataResult	false	The device status data subscription status. See section 11.6.3.2 VehicleDataResult (VEHICLEDATA_DEVICESTATUS must be used).		

Param Name	Type	Mandatory	Description
driverBraking	Common.VehicleDataResult	false	The brake pedal status data subscription status. See section 11.6.3.2 <i>VehicleDataResult</i> (VEHICLEDATA_BRAKING must be used).
wiperStatus	Common.VehicleDataResult	false	The wipers status data subscription status. See section 11.6.3.2 <i>VehicleDataResult</i> (VEHICLEDATA_WIPERSTATUS must be used).
headLampStatus	Common.VehicleDataResult	false	The head lamps status data subscription status. See section 11.6.3.2 <i>VehicleDataResult</i> (VEHICLEDATA_HEADLAMPSTATUS must be used).
engineTorque	Common.VehicleDataResult	false	Torque value for engine data subscription status. See section 11.6.3.2 <i>VehicleDataResult</i> (VEHICLEDATA_ENGINETORQUE must be used).
accPedalPosition	Common.VehicleDataResult	false	Accelerator pedal position data subscription status. See section 11.6.3.2 <i>VehicleDataResult</i> (VEHICLEDATA_ACCPEDAL must be used).
steeringWheelAngle	Common.VehicleDataResult	false	Current angle of the steering wheel data subscription status. See section 11.6.3.2 <i>VehicleDataResult</i> (VEHICLEDATA_STEERINGWHEEL must be used).

11.6.3.2 *VehicleDataResult*

Param Name	Type	Mandatory	Description
dataType	Common.VehicleDataType	true	The data type being subscribed. Must correspond to the name of related parameter from section 11.6.3.1.
resultCode	Common.VehicleDataResultCode	true	Result code that defines the subscription status for the named data type.

11.6.3.3 *VehicleDataType Enumeration*

Element name	Value	Short Description
VEHICLEDATA_GPS	0	Parameter that must contain this value: gps
VEHICLEDATA_SPEED	1	Parameter that must contain this value: speed
VEHICLEDATA_RPM	2	Parameter that must contain this value: rpm
VEHICLEDATA_FUELLEVEL	3	Parameter that must contain this value: fuelLevel
VEHICLEDATA_FUELLEVEL_STATE	4	Parameter that must contain this value: fuelLevel_State
VEHICLEDATA_FUELCONSUMPTION	5	Parameter that must contain this value: instantFuelConsumption
VEHICLEDATA_EXTERNTTEMP	6	Parameter that must contain this value: externalTemperature

Element name	Value	Short Description
VEHICLEDATA_VIN	7	Not used for subscription
VEHICLEDATA_PRNDL	8	Parameter that must contain this value: prndl
VEHICLEDATA_TIREPRESSURE	9	Parameter that must contain this value: tirePressure
VEHICLEDATA_ODOMETER	10	Parameter that must contain this value: odometer
VEHICLEDATA_BELTSTATUS	11	Parameter that must contain this value: beltStatus
VEHICLEDATA_BODYINFO	12	Parameter that must contain this value: bodyInformation
VEHICLEDATA_DEVICESSTATUS	13	Parameter that must contain this value: deviceStatus
VEHICLEDATA_BRaking	19	Parameter that must contain this value: driverBraking
VEHICLEDATA_WIPERSTATUS	20	Parameter that must contain this value: wiperStatus
VEHICLEDATA_HEADLAMPSTATUS	21	Parameter that must contain this value: headLampStatus
VEHICLEDATA_BATTVOLTAGEx	22	Not used for subscription
VEHICLEDATA_ENGINETORQUE	23	Parameter that must contain this value: engineTorque
VEHICLEDATA_ACCPEDAL	24	Parameter that must contain this value: accPedalPosition
VEHICLEDATA_STEERINGWHEEL	25	Parameter that must contain this value: steeringWheelAngle

**FORD specific VehicleDataType Enumeration
(extension of 11.6.3.3 table)**

VEHICLEDATA_ECALLINFO	1 4	Parameter that must contain this value: eCallInfo
VEHICLEDATA_AIRBAGSTATUS	1 5	Parameter that must contain this value: airbagStatus
VEHICLEDATA_EMERGENCYEVENT	1 6	Parameter that must contain this value: emergencyEvent

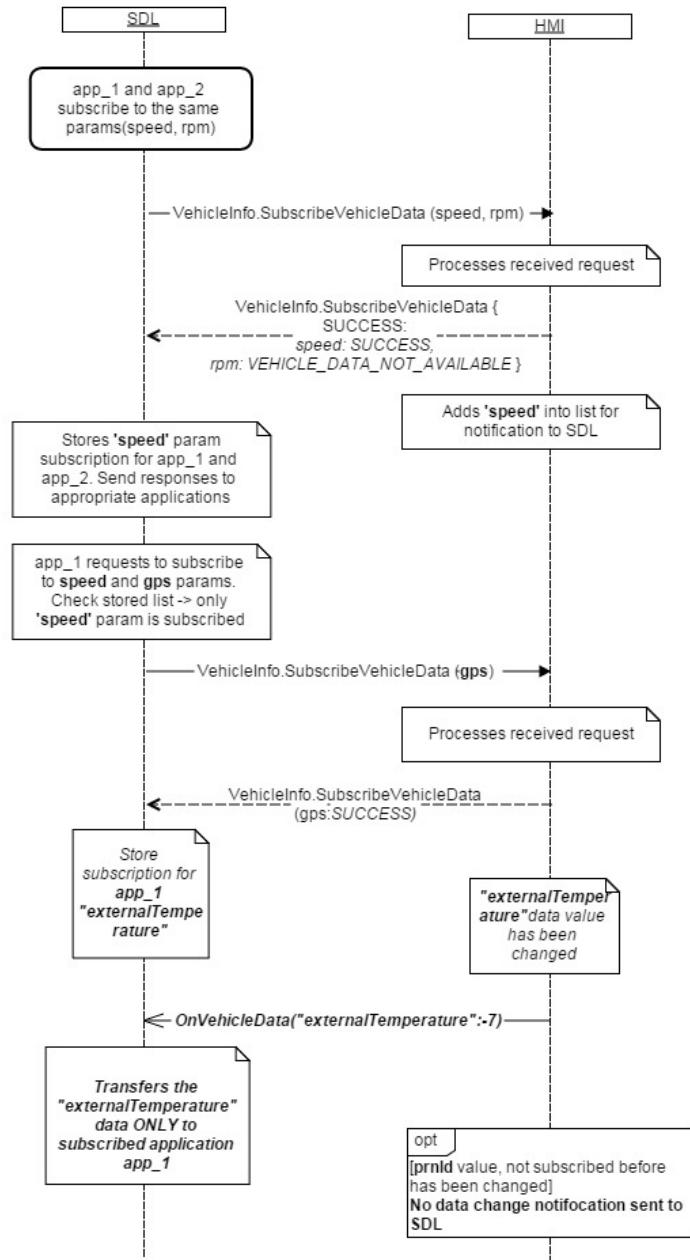
		nt
VEHICLEDATA_CLUSTERMODESTATUS	17	Parameter that must contain this value: clusterModes
VEHICLEDATA_MYKEY	18	Parameter that must contain this value: myKey
VEHICLEDATA_ECALLINFO	14	Parameter that must contain this value: eCallInfo

11.6.3.4 VehicleDataResultCode Enumeration

Element name	Value	Short Description
SUCCESS	0	The named data type is successfully subscribed. Whenever it receives the updates, HMI will notify SDL about the event via OnVehicleData.
TRUNCATED_DATA	1	Not applicable.
DISALLOWED	2	Not applicable.
USER_DISALLOWED	3	Providing periodic updates for the named data type is disallowed by the User.
INVALID_ID	4	Not applicable.
VEHICLE_DATA_NOT_AVAILABLE	5	The named data type is not supported (or reported, published) by HMI.
DATA_ALREADY_SUBSCRIBED	6	The named data type is already marked by HMI to receive periodic update notifications.
DATA_NOT_SUBSCRIBED	7	Not applicable.
IGNORED	8	Not applicable.

11.6.4 Sequence Diagrams

11.6.4.1 SubscribeVehicleData



11.6.5 JSON Messages Examples

11.6.5.1 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,
    }
}

```

11.6.5.2 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 : HEADLAMPSTATUS,
        resultCode : SUCCESS
} ,

"accPedalPosition" :
{
    dataType :
VEHICLEDATA_ACCPEDAL,
        resultCode :
VEHICLE_DATA_NOT_AVAILABLE
} ,

"code" : 0,
"method" :
"VehicleInfo.SubscribeVehicleData"
}
}
```

11.6.5.3 Error message

```
{  
    "id" : 139,  
    "jsonrpc" : "2.0",  
    "error" :  
    {  
        "code" : 6,  
        "message" : "All of requested data  
types is subscribed already",  
        "data" :  
        {  
            "method" :  
"VehicleInfo.SubscribeVehicleData"  
        }  
    }  
}
```

11.7 UnSubscribeVehicleData

11.7.1 Description

Type:	Function
Sender:	SDL
Purpose:	Unsubscribe from definite vehicle data type(s) updates

Initially SDL requests HMI to provide periodic updates for the named data types via `SubscribeVehicleData`. Once HMI responds successfully (see section 11.6 `SubscribeVehicleData`), it shall inform SDL upon every data update of the named data type via `OnVehicleData` notification.

Via `UnsubscribeVehicleData` SDL requests HMI to stop sending updates for the named data type(s) previously subscribed.

The request may come for the application being whatever active or in background on HMI (depends on SDL Policy Manager permissions).

11.7.2 Request

11.7.2.1 Behavior

HMI must:

- 1) Check whether the named data type may be successfully unsubscribed. The data types are defined via Boolean type parameters to be unsubscribed (`gps`, `speed`, `rpm` and other that are provided in *section 11.7.2.2*):
Parameters of `true` value must to be unsubscribed
One or several or all of the parameters previously subscribed may be included into request (see 11.7.2.2).

Note:

Within a request SDL must provide at least one ‘true’-value parameter of data type to be unsubscribed.

- 2) Delete the requested data type from the internal list of subscribed items which get OnVehicleData update notifications for a correspondant data type.
- 3) Respond with one of appropriate result codes (see section 11.7.3 Response). And in case of SUCCESS return the parameter(s) of the name(s) equal to those requested for subscription. The returned parameters must contain Data type (see section 11.7.3.3) correspondent to the name of related parameter (for example,
`gps : {VEHICLEDATA_GPS, SUCCESS},
speed : {VEHICLEDATA_SPEED,
SUCCESS}).`
And individual VehicleDataResult code from section 11.7.3.4. The codes different from SUCCESS will inform SDL about the status of the named data type and whether it is unsubscribed indeed.
- 4) Stop providing OnVehicleData notifications upon any changes of successfully unsubscribed vehicle data type.

SDL Important Note: HMI is not being provided with the information which one application requests the data unsubscription, SDL manages this internally and redistributes the data received via OnVehicleData between the subscribed applications by itself. Thus, unsubscription is requested on HMI only if there's no application subscribed on this data type at all.

When an application disconnects unexpectedly, SDL sends UnsubscribeVehicleData request for all VehicledataTypes subscribed only by the application just disconnected. See [11.7.4.2 UnSubscribeVehicleData unexpected disconnect](#)

SDL 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

11.7.2.2 Parameters

Param Name	Type	Mandatory	Description
gps	Boolean	false	Unsubscribe from GPS data updates. Information related to GPS data: number of satellites, compass direction, longitude, latitude, etc. See GPSData
speed	Boolean	false	Unsubscribe from vehicle speed updates that are provided in kilometers per hour.
rpm	Boolean	false	Unsubscribe from the number of engine revolutions per minute updates.
fuelLevel	Boolean	false	Unsubscribe from the fuel level in the tank updates (percentage).

Param Name	Type	Mandatory	Description
fuelLevel_State	Boolean	false	Unsubscribe from the fuel level state updates: normal, low, etc. See ComponentVolumeStatus .
instantFuelConsumption	Boolean	false	Unsubscribe from the instantaneous fuel consumption updates to be provided in microliters.
externalTemperature	Boolean	false	Unsubscribe from the external temperature that is provided in degrees Celsius.
prndl	Boolean	false	Unsubscribe from gear stick position updates: first, second, etc. See PRNDL_ .
tirePressure	Boolean	false	Unsubscribe from tire pressure status updates: - The warning status (on, off, etc.) - The status of the tire pressure itself: normal, low, etc. See TireStatus .
odometer	Boolean	false	Unsubscribe from the information from odometer that is provided in km.
beltStatus	Boolean	false	Unsubscribe from the information of the seat belts status updates: deployed, buckled. See BeltStatus .
bodyInformation	Boolean	false	Unsubscribe from updates of body information including power modes: park brake status, ignition and ignition stable status. See BodyInformation .
deviceStatus	Boolean	false	Unsubscribe from updates of device status including signal and battery strength. See DeviceStatus .
driverBraking	Boolean	false	Unsubscribe from the information of the brake pedal status updates: on, off, etc. See VehicleDataEventStatus .
wiperStatus	Boolean	false	Unsubscribe from updates of the wipers status: when they are manually on, manually off, stalled, etc. See WiperStatus .
headLampStatus	Boolean	false	Unsubscribe from the updates of the head lamps status: when they are on, off, etc. See HeadLampStatus .
engineTorque	Boolean	false	Unsubscribe from the updates of the torque value for engine (in Nm) on non-diesel variants
accPedalPosition	Boolean	false	Unsubscribe from the information of accelerator pedal position (percentage depressed).
steeringWheelAngle	Boolean	false	Unsubscribe from the information of current angle of the steering wheel (in degrees)

11.7.3 Response

Note:

There is a difference in message type for WebSocket and D-Bus connection (see the table below).

Result	Description	Message type		Message Params	Notes
		WebSocket	D-Bus		
Success	SUCCESS: HMI processes the request successfully and returns the information about the data type(s) and the corresponding individual unsubscription result code (see p.3 of section 11.7.2.1)	JSON response	Method return	Parameters of the names equal to requested ones, code: 0	See 11.7.3.1 Parameters.

Failure	IGNORED: Unsubscribe is requested for the single data not yet subscribed			Parameters of the names equal to requested ones, Code : 6	Applicable for this RPC result codes. Please see Result Enumeration for all SDL-supported codes.
	DATA_NOT_AVAILABLE: All of requested data types did not and will not receive update notifications since they are not supported (or reported, published) by HMI.			code : 9	
	INVALID_ID Wrong appID	JSON error message	Method return	code : 13	
	GENERIC_ERROR 1) The unknown issue occurred or other codes are not applicable.			code : 22	

11.7.3.1 Parameters

Param Name	Type	Mandatory	Description	
gps	Common.VehicleDataResult	false	GPS data unsubscription status. See section 11.7.3.2 VehicleDataResult (VEHICLEDATA_GPS must be used).	
speed	Common.VehicleDataResult	false	Vehicle speed unsubscription status. See section 11.7.3.2 VehicleDataResult (VEHICLEDATA_SPEED must be used).	
rpm	Common.VehicleDataResult	false	The engine number of revolutions per minute data unsubscription status. See section 11.7.3.2 VehicleDataResult (VEHICLEDATA_RPM must be used).	
fuelLevel	Common.VehicleDataResult	false	The fuel level data unsubscription status. See section 11.7.3.2 VehicleDataResult (VEHICLEDATA_FUELLEVEL must be used).	
fuelLevel_State	Common.VehicleDataResult	false	The fuel level state data unsubscription status. See section 11.7.3.2 VehicleDataResult (VEHICLEDATA_FUELLEVEL_STATE must be used).	
instantFuelConsumption	Common.VehicleDataResult	false	The instantaneous fuel consumption data unsubscription status. See section 11.7.3.2 VehicleDataResult (VEHICLEDATA_FUELCONSUMPTION must be used).	
externalTemperature	Common.VehicleDataResult	false	The external temperature data unsubscription status. See section 11.7.3.2 VehicleDataResult (VEHICLEDATA_EXTERNTEMP must be used).	
prndl	Common.VehicleDataResult	false	The gear stick position data unsubscription status. See section 11.7.3.2 VehicleDataResult (VEHICLEDATA_PRNDL must be used).	
tirePressure	Common.VehicleDataResult	false	The tire pressure data unsubscription status. See section 11.7.3.2 VehicleDataResult (VEHICLEDATA_TIREPRESSURE must be used).	
odometer	Common.VehicleDataResult	false	Odometer data unsubscription status. See section 11.7.3.2 VehicleDataResult (VEHICLEDATA_ODOMETER must be used).	
beltStatus	Common.VehicleDataResult	false	The seat belts status data unsubscription status. See section 11.7.3.2 VehicleDataResult	

Param Name	Type	Mandatory	Description
			(VEHICLEDATA_BELTSTATUS must be used).
bodyInformation	Common.VehicleDataResult	false	The body information data unsubscription status. See section 11.7.3.2 VehicleDataResult (VEHICLEDATA_BODYINFO must be used).
deviceStatus	Common.VehicleDataResult	false	The device status data unsubscription status. See section 11.7.3.2 VehicleDataResult (VEHICLEDATA_DEVICESTATUS must be used).
driverBraking	Common.VehicleDataResult	false	The brake pedal status data unsubscription status. See section 11.7.3.2 VehicleDataResult (VEHICLEDATA_BRAKING must be used).
wiperStatus	Common.VehicleDataResult	false	The wipers status data unsubscription status. See section 11.7.3.2 VehicleDataResult (VEHICLEDATA_WIPERSTATUS must be used).
headLampStatus	Common.VehicleDataResult	false	The head lamps status data unsubscription status. See section 11.7.3.2 VehicleDataResult (VEHICLEDATA_HEADLAMPSTATUS must be used).
engineTorque	Common.VehicleDataResult	false	Torque value for engine data unsubscription status. See section 11.7.3.2 VehicleDataResult (VEHICLEDATA_ENGINETORQUE must be used).
accPedalPosition	Common.VehicleDataResult	false	Accelerator pedal position data sunscription status. See section 11.7.3.2 VehicleDataResult (VEHICLEDATA_ACCPEDAL must be used).
steeringWheelAngle	Common.VehicleDataResult	false	Current angle of the steering wheel data unsubscription status. See section 11.7.3.2 VehicleDataResult (VEHICLEDATA_STEERINGWHEEL must be used).

11.7.3.2 VehicleDataResult

Param Name	Type	Mandatory	Description
dataType	Common.VehicleDataType	true	The data type being unsubscribed. Must correspond to the name of related parameter from section 11.7.3.1.
resultCode	Common.VehicleDataResultCode	true	Result code that defines the unsubscription status for the named data type.

11.7.3. VehicleDataType Enumeration

Element name	Value	Short Description
VEHICLEDATA_GPS	0	Parameter that must contain this value: gps
VEHICLEDATA_SPEED	1	Parameter that must contain this value: speed
VEHICLEDATA_RPM	2	Parameter that must contain this value: rpm
VEHICLEDATA_FUELLEVEL	3	Parameter that must contain this value: fuelLevel
VEHICLEDATA_FUELLEVEL_STATE	4	Parameter that must contain this value: fuelLevel_State

Element name	Value	Short Description
VEHICLEDATA_FUELCONSUMPTION	5	Parameter that must contain this value: instantFuelConsumption
VEHICLEDATA_EXTERNTEMP	6	Parameter that must contain this value: externalTemperature
VEHICLEDATA_VIN	7	Not used for unsubscription
VEHICLEDATA_PRNDL	8	Parameter that must contain this value: prndl
VEHICLEDATA_TIREPRESSURE	9	Parameter that must contain this value: tirePressure
VEHICLEDATA_ODOMETER	10	Parameter that must contain this value: odometer
VEHICLEDATA_BELTSTATUS	11	Parameter that must contain this value: beltStatus
VEHICLEDATA_BODYINFO	12	Parameter that must contain this value: bodyInformation
VEHICLEDATA_DEVICESSTATUS	13	Parameter that must contain this value: deviceStatus
VEHICLEDATA_BRKING	19	Parameter that must contain this value: driverBraking
VEHICLEDATA_WIPERSTATUS	20	Parameter that must contain this value: wiperStatus
VEHICLEDATA_HEADLAMPSTATUS	21	Parameter that must contain this value: headLampStatus
VEHICLEDATA_BATTVOLTAG	22	Not used for unsubscription
VEHICLEDATA_ENGINETORQUE	23	Parameter that must contain this value: engineTorque
VEHICLEDATA_ACCPEDAL	24	Parameter that must contain this value: accPedalPosition
VEHICLEDATA_STEERINGWHEEL	25	Parameter that must contain this value: steeringWheelAngle

***FORD specific VehicleDataType Enumeration
(extension of 11.7.3.3 table)***

VEHICLEDATA_ECALLINFO	14	Parameter that must contain this value:
-----------------------	----	---

		eCallInfo
VEHICLEDATA_AIRBAGSTATUS	1 5	Parameter that must contain this value: airbagStatus
VEHICLEDATA_EMERGENCYEVENT	1 6	Parameter that must contain this value: emergencyEvent
VEHICLEDATA_CLUSTERMODESTATUS	1 7	Parameter that must contain this value: clusterModes
VEHICLEDATA_MYKEY	1 8	Parameter that must contain this value: myKey
VEHICLEDATA_ECALLINFO	1 4	Parameter that must contain this value: eCallInfo

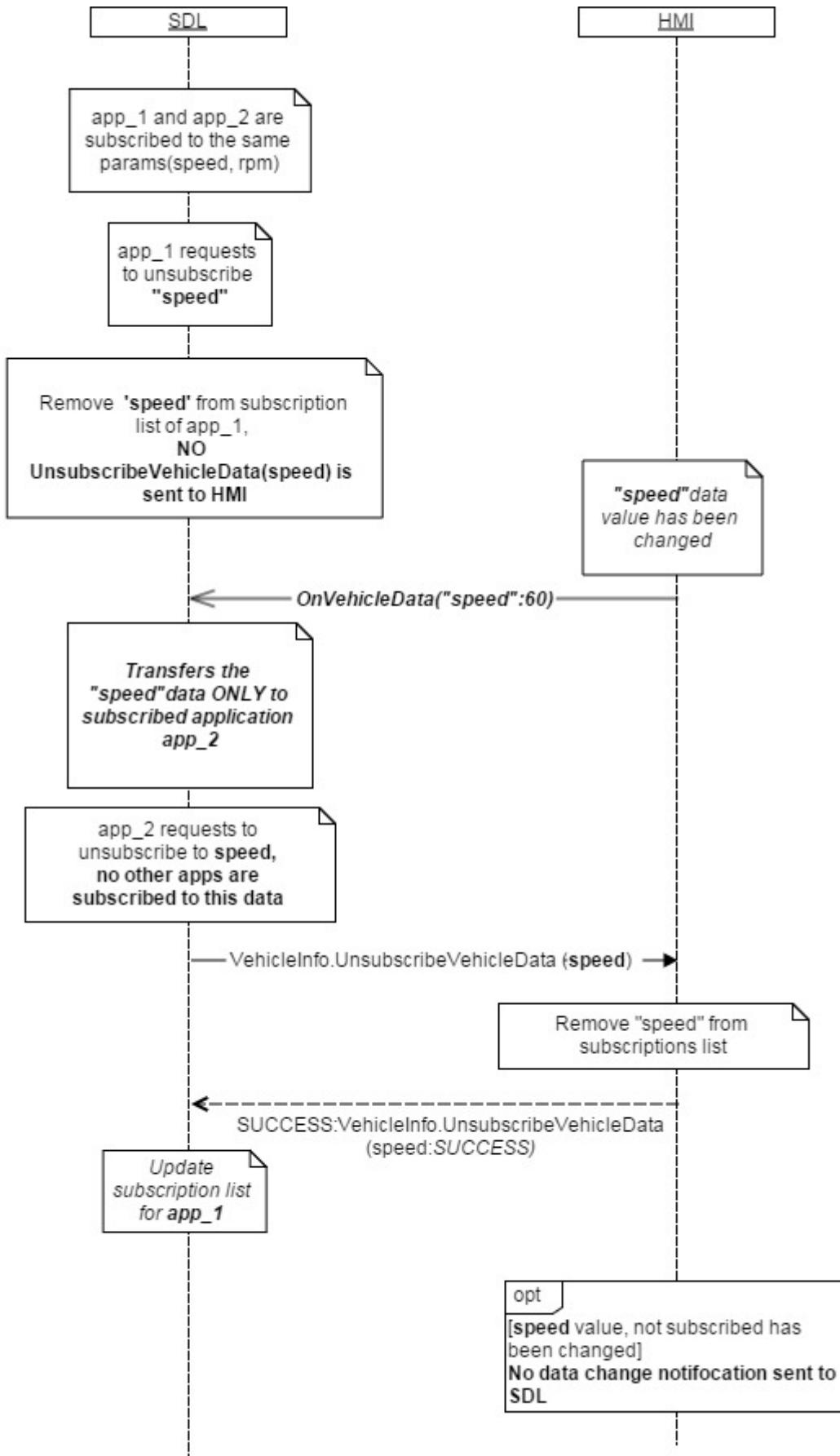
11.7.3. VehicleDataresultCode Enumeration

Element name	Value	Short Description
SUCCESS	0	The named data type is successfully unsubscribed. HMI shall NOT notify SDL about any changes of this data type further more.
TRUNCATED_DATA	1	Not applicable.
DISALLOWED	2	Not applicable.
USER_DISALLOWED	3	Not applicable.
INVALID_ID	4	Not applicable.
VEHICLE_DATA_NOT_AVAILABLE	5	The named data type did not and will not receive update notifications since it is not supported (or reported, published) by HMI.

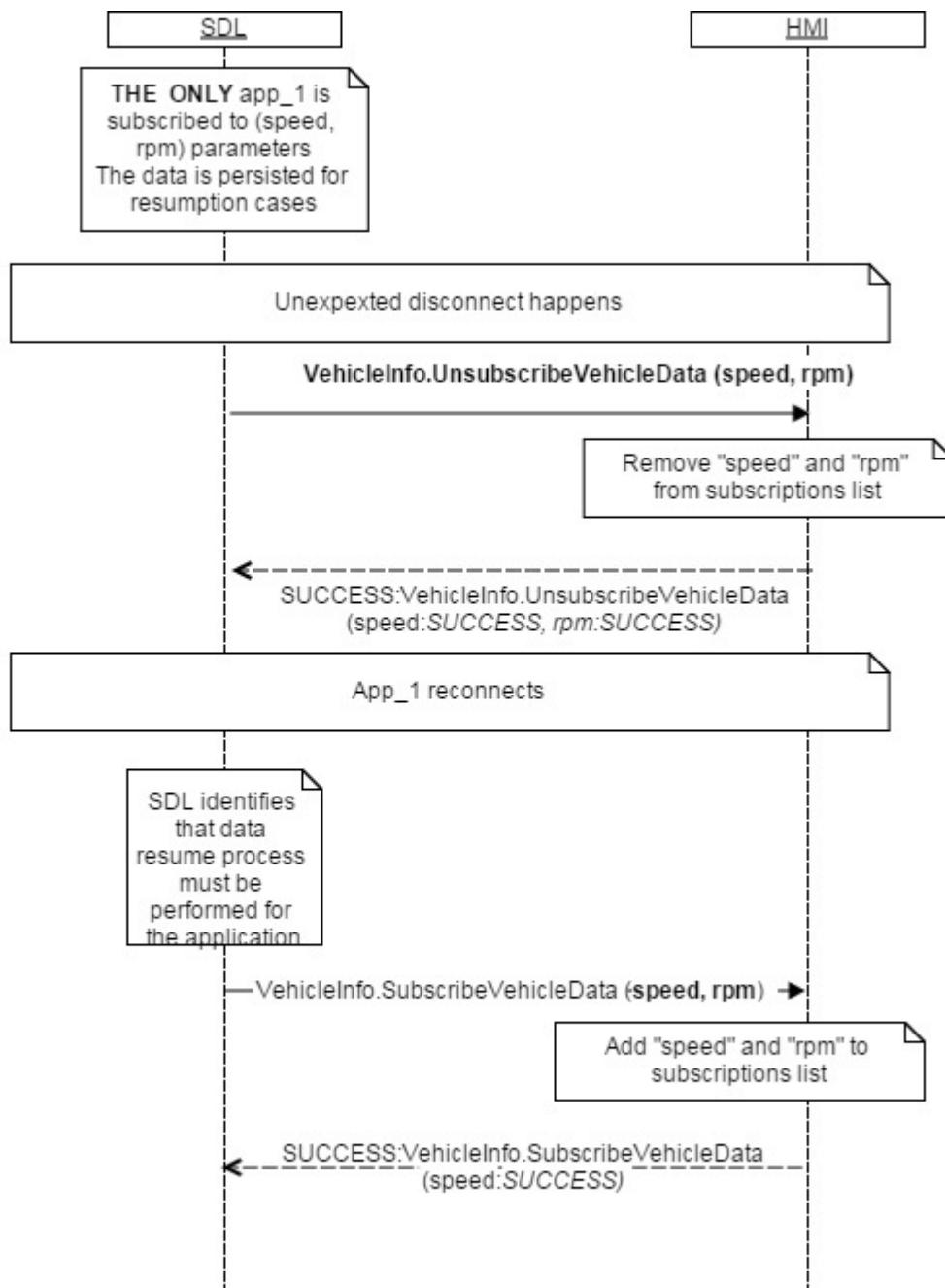
Element name	Value	Short Description
DATA_ALREADY_SUBSCRIBED	6	Not applicable
DATA_NOT_SUBSCRIBED	7	The named data type cannot be unsubscribed because it has not been subscribed yet.
IGNORED	8	Not applicable.

11.7.4 Sequence Diagrams

11.7.4.1 *UnSubscribeVehicleData*



11.7.4.2 UnSubscribeVehicleData unexpected disconnect



11.7.5 JSON Messages Examples

11.7.5.1 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,
    }
}

```

11.7.5.2 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 : HEADLAMPSTATUS,
                resultCode : SUCCESS
        } ,


        "accPedalPosition" :
        {
            dataType :
VEHICLEDATA_ACCPEDAL,
                resultCode :
DATA_NOT_SUBSCRIBED
        } ,


        "code" : 0,
        "method" :
"VehicleInfo.UnsubscribeVehicleData"
    }
}
```

11.7.5.3 Error message

```
{  
    "id" : 139,  
    "jsonrpc" : "2.0",  
    "error" :  
    {  
        "code" : 22,  
        "message" : "An unknown error  
occurred",  
        "data" :  
        {  
            "method" :  
"VehicleInfo.UnsubscribeVehicleData"  
        }  
    }  
}
```

11.8 GetVehicleData

11.8.1 Description

Type:	Function
Sender:	SDL
Purpose:	Receive current values of vehicle data

SDL sends `GetVehicleData` (in contrast to `SubscribeVehicleData`) for getting the current value(s) of the named data type(s) without subscription on it.

11.8.2 Request

11.8.2.1 Behavior

HMI must:

1. Check whether the current values of requested data type can be provided. The data type is defined with Boolean parameters of `gps`, `speed`, `rpm` and other that are provided in section 11.8.2.2:

- true value parameters are those which require information in response.
- One or several or all of the parameters from section 11.8.2.2 may be included into request.

Note:

Within a request SDL will provide at least one 'ture'-value parameter of data type the current values of which must be provided.

2. Respond with one of appropriate result codes (see section 11.8.3 Response).
 - 2.1. HMI must return the parameter(s) of the name(s) equal to those requested, only together with result codes of:
 - `SUCCESS`: when HMI can provide the current values of all requested data types.
 - `SUCCESS` when HMI can provide just one or several values of the total requested (that is, less

than requested). The example cases: when sensor is broken or HMI does not support or publish some data type.

See the table below for clarification.

Request	Response		
	code: SUCCESS	code: SUCCESS	code: SUCCESS
	data available: all	data available: some	data available: none
data_type_1 data_type_2 data_type_3 data_type_4 data_type_5	data_type_1 data_type_2 data_type_3 data_type_4 data_type_5	data_type_1 data_type_3 data_type_4	

2.2. Every of parameters returned must contain the appropriate values as described in section 11.8.3.1.

11.8.2.2 Parameters

Param Name	Type	Mandatory	Description
gps	Boolean	false	Get GPS data current values. Information related to GPS data: number of satellites, compass direction, longitude, latitude, etc. See GPSData
speed	Boolean	false	Get vehicle speed current value to be provided in kilometers per hour.
rpm	Boolean	false	Get the number of engine revolutions per minute current value.
fuelLevel	Boolean	false	Get the fuel level in the tank current value (percentage).
fuelLevel_State	Boolean	false	Get the fuel level state current value: normal, low, etc. See ComponentVolumeStatus .
instantFuelConsumption	Boolean	false	Get the instantaneous fuel consumption current value to be provided in microliters.
externalTemperature	Boolean	false	Get the external temperature current value to be provided in degrees Celsius.
vin	Boolean	false	Vehicle identification number.
prndl	Boolean	false	Get the gear stick position current value: first, second, etc. See PRNDL .
tirePressure	Boolean	false	Get tire pressure status current values: - The warning status (on, off, etc.) - The status of the tire pressure itself: normal, low, etc. See TireStatus .
odometer	Boolean	false	Get the current information from odometer to be provided in km.
beltStatus	Boolean	false	Get the current information of the seat belts status: deployed, buckled. See BeltStatus .
bodyInformation	Boolean	false	Get the current values of body information including power modes: park brake status, ignition and ignition stable status. See BodyInformation .
deviceStatus	Boolean	false	Get the current values of device status including signal and battery strength. See DeviceStatus .
driverBraking	Boolean	false	Get the current information of the brake pedal status updates: on, off, etc. See VehicleDataEventStatus .

Param Name	Type	Mandatory	Description
wiperStatus	Boolean	false	Get the current information of the wipers status: whether they are manually on, manually off, stalled, etc. See WiperStatus .
headLampStatus	Boolean	false	Get the current information of the head lamps status: when they are on, off, etc. See HeadLampStatus .
engineTorque	Boolean	false	Get the current information of the torque value for engine (in Nm) on non-diesel variants
accPedalPosition	Boolean	false	Get the current information of accelerator pedal position (percentage depressed).
steeringWheelAngle	Boolean	false	Get the current information of current angle of the steering wheel (in degrees)
appID	Integer	true	Id of application that requested this RPC.

11.8.3 Response

Note:

There is a difference in message type for WebSocket and D-Bus connection (see the table below).

Result	Description	Message type		Message Params	Notes
		WebSocket	D-Bus		
Success	SUCCESS: HMI provides values of all requested vehicle data.	JSON response	Method return	Parameters of the names equal to requested ones, code: 0	See 11.8.3.1 Parameters .
Failure	DATA_NOT_AVAILABLE: <u>1.</u> HMI can provide just one or several values of the total requested (that is, less than requested).	JSON error message	Method return	Parameters of the names equal to requested ones, code: 9	See 11.8.2.1 Behavior .
	DATA_NOT_AVAILABLE: <u>2.</u> HMI can provide none of the requested data types.			code: 9	Applicable for this RPC result codes. Please see Result Enumeration for all SDL-supported codes.
	INVALID_ID appID is invalid			code: 13	
	GENERIC_ERROR: The unknown issue occurred or other codes are not applicable.			code: 22	

11.8.3.1 Parameters

Param Name	Type	Mandatory	Additional	Description
gps	Common.GPSData	false	-	The current values of GPS data: number of satellites, compass direction, longitude, latitude, etc. See section 11.8.3.2 GPSData
speed	Float	false	minvalue = 0 maxvalue = 700	The current value of vehicle speed in kilometers per hour.

Param Name	Type	Mandatory	Additional	Description
rpm	Integer	false	minvalue = 0 maxvalue = 20000	The current number of revolutions per minute of the engine.
fuelLevel	Float	false	minvalue = -6 maxvalue = 106	The current value of fuel level in the tank (percentage).
fuelLevel_State	Common.ComponentVolumeStatus	false	-	The current value of fuel level state: normal, low, etc. See ComponentVolumeStatus
instantFuelConsumption	Float	false	minvalue = 0 maxvalue = 25575	The current value of instantaneous fuel consumption in microliters.
externalTemperature	Float	false	minvalue = -40 maxvalue = 100	The current external temperature in degrees Celsius.
vin	String	false	maxlength = 17	Vehicle identification number.
prndl	Common.PRNDL	false	-	The current position of gear stick: first, second, etc. See PRNDL.
tirePressure	Common.TireStatus	false	-	The current information of: - The warning status (on, off, etc.) - The status of the tire pressure itself: normal, low, etc. See TireStatus.
odometer	Integer	false	minvalue = 0 maxvalue = 17000000	Current value of odometer in km.
beltStatus	Common.BeltStatus	false	-	The current status of the seat belts: deployed, buckled. See BeltStatus
bodyInformation	Common.BodyInformation	false	-	The cuurent body information: park brake status, ignition and ignition stable status. See BodyInformation
deviceStatus	Common.DeviceStatus	false	-	The current information of device status including signal and battery strength. See DeviceStatus.
driverBraking	Common.VehicleDataEventStatus	false	-	The current status of the brake pedal: on, off, etc. See VehicleDataEventStatus
wiperStatus	Common.WiperStatus	false	-	The current status of the wipers: whether they are manually on, manually off, stalled, etc. See WiperStatus
headLampStatus	Common.HeadLampStatus	false	-	The current status of the head lamps: whether they are on, off, etc. See HeadLampStatus
engineTorque	Float	false	minvalue = -1000 maxvalue = 2000	The current torque value for engine (in Nm) for non-diesel models.

Param Name	Type	Mandatory	Additional	Description
accPedalPosition	Float	false	minvalue = 0 maxvalue = 100	The current value of accelerator pedal position (percentage depressed).
steeringWheelAngle	Float	false	minvalue = -2000 maxvalue = 2000	Current angle of the steering wheel (in degrees).

11.8.3.2 GPSData

Param Name	Type	Mandatory	Maxvalue	Description
longitudeDegrees	Float	false	minvalue = -180 maxvalue = 180	Position longitude in degrees.
latitudeDegrees	Float	false	minvalue = -90 maxvalue = 90	Position latitude in degrees.
utcYear	Integer	false	minvalue = 2010 maxvalue = 2100	The current UTC year.
utcMonth	Integer	false	minvalue = 1 maxvalue = 12	The current UTC month.
utcDay	Integer	false	minvalue = 1 maxvalue = 31	The current UTC day.
utcHours	Integer	false	minvalue = 0 maxvalue = 23	The current UTC hour.
utcMinutes	Integer	false	minvalue = 0 maxvalue = 59	The current UTC minute.
utcSeconds	Integer	false	minvalue = 0 maxvalue = 59	The current UTC second.
compassDirection	Common.CompassDirection	false	-	The compass direction: north, east, etc. See section 11.8.3.3 CompassDirection
pdop	Float	false	minvalue = 0 maxvalue = 10	Positional Dilution Of Precision.
hdop	Float	false	minvalue = 0 maxvalue = 10	Horizontal Dilution Of Precision.
vdop	Float	false	minvalue = 0 maxvalue = 10	Vertical Dilution Of Precision.
actual	Boolean	false	-	The information about actuality of GPS data: 'true', if actual 'false', if inferred
satellites	Integer	false	minvalue = 0 maxvalue = 31	Number of satellites in view.
dimension	Common.Dimension	false	-	The supported GPS dimension: 2D, 3D. See section 11.8.3.4 Dimension
altitude	Float	false	minvalue = -10000 maxvalue =	Altitude in meters.

Param Name	Type	Mandatory	Maxvalue	Description
			10000	
heading	Float	false	minvalue = 0 maxvalue = 359.99	'0' is considered as heading North. SDL expects to receive data with resolution of 0.01.
speed	Float	false	minvalue = 0 maxvalue = 500	The speed in KPH.

11.8.3.3 CompassDirection

Element name	Value	Short Description
NORTH	0	Represents the North compass direction
NORTHWEST	1	Represents the North-West compass direction
WEST	2	Represents the West compass direction
SOUTHWEST	3	Represents the South-West compass direction
SOUTH	4	Represents the South compass direction
SOUTHEAST	5	Represents the South-East compass direction
EAST	6	Represents the East compass direction
NORTHEAST	7	Represents the North-East compass direction

11.8.3.4 Dimension

Element name	Value	Short Description
NO_FIX	0	No GPS at all
2D	1	Longitude and latitude
3D	2	Longitude and latitude and altitude

11.8.3.5 ComponentVolumeStatus

Element name	Value	Short Description
UNKNOWN	0	The data is unknown.
NORMAL	1	The volume is normal.
LOW	2	The volume is low.
FAULT	3	The module/sensor is currently faulted.
ALERT	4	The component's volume is in critical level.
NOT_SUPPORTED	5	The data is not supported.

11.8.3.6 PRNDL

Element name	Value	Short Description
PARK	0	Parking
REVERSE	1	Reverse gear
NEUTRAL	2	No gear
DRIVE	3	Drive Sport mode
SPORT	4	Sport mode

Element name	Value	Short Description
LOWGEAR	5	1st gear hold
FIRST	6	1st gear
SECOND	7	2nd gear
THIRD	8	3d gear
FOURTH	9	4th gear
FIFTH	10	5th gear
SIXTH	11	6th gear
SEVENTH	12	7th gear
EIGHTH	13	8th gear
FAULT	14	The module/sensor is currently faulted.

11.8.3.7 TireStatus

Param Name	Type	Mandatory	Description
pressureTelltales	Common.WarningLightStatus	false	Status of the Tire Pressure Telltale. See WarningLightStatus.
leftFront	Common.SingleTireStatus	false	The status of the left front tire.
rightFront	Common.SingleTireStatus	false	The status of the right front tire.
leftRear	Common.SingleTireStatus	false	The status of the left rear tire.
rightRear	Common.SingleTireStatus	false	The status of the right rear tire.
innerLeftRear	Common.SingleTireStatus	false	The status of the inner left rear.
innerRightRear	Common.SingleTireStatus	false	The status of the inner right rear.

11.8.3.8 WarningLightStatus

Element name	Value	Short Description
OFF	0	The warning light is off
ON	1	The warning light is on
FLASH	2	The warning light is flashing
NOT_USED	3	There is no warning light for the event on HU.

11.8.3.9 SingleTireStatus

Param Name	Type	Mandatory	Description
status	Common.ComponentVolumeStatus	true	See section 11.8.3.5 ComponentVolumeStatus

11.8.3.10 BeltStatus

Param Name	Type	Mandatory	Description
driverBeltDeployed	Common.VehicleDataEventStatus	false	The driver seat belt is deployed.
passengerBeltDeployed	Common.VehicleDataEventStatus	false	The passenger seat belt is deployed.

Param Name	Type	Mandatory	Description
passengerBuckleBelted	Common.VehicleDataEventStatus	false	The passenger seat belt is buckled.
driverBuckleBelted	Common.VehicleDataEventStatus	false	The driver seat belt is buckled.
leftRow2BuckleBelted	Common.VehicleDataEventStatus	false	The left seat belt of the 2 nd row is buckled.
passengerChildDetected	Common.VehicleDataEventStatus	false	The child passenger is detected.
rightRow2BuckleBelted	Common.VehicleDataEventStatus	false	The right seat belt of the 2 nd row is buckled.
middleRow2BuckleBelted	Common.VehicleDataEventStatus	false	The middle seat belt of the 2 nd row is buckled.
middleRow3BuckleBelted	Common.VehicleDataEventStatus	false	The middle seat belt of the 3 rd row is buckled.
leftRow3BuckleBelted	Common.VehicleDataEventStatus	false	The left seat belt of the 3 rd row is buckled.
rightRow3BuckleBelted	Common.VehicleDataEventStatus	false	The right seat belt of the 3 rd row is buckled.
leftRearInflatableBelted	Common.VehicleDataEventStatus	false	The left rear inflatable is belted.
rightRearInflatableBelted	Common.VehicleDataEventStatus	false	The right rear inflatable is belted.
middleRow1BeltDeployed	Common.VehicleDataEventStatus	false	The seat belt of the middle row is deployed.
middleRow1BuckleBelted	Common.VehicleDataEventStatus	false	The seat belt of the middle row is buckled.

11.8.3.13 VehicleDataEventStatus

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

Element name	Value	Short Description
NO_EVENT	0	The system does not have the adequate information to send valid YES or NO states.
NO	1	The requested event is in NO state
YES	2	The requested event is in YES state.
NOT_SUPPORTED	3	The requested data is not supported

11.8.3.15 EmergencyEventType

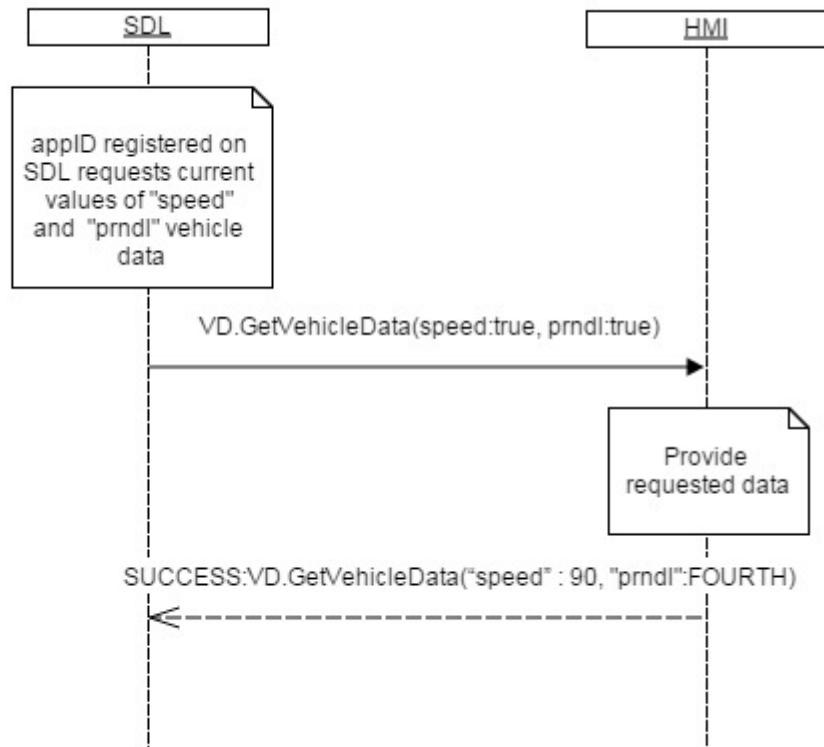
Reflects the emergency event status of the vehicle.

Element name	Short Description
NO_EVENT	References signal "VedsEvtType_D_Ltchd". See EmergencyEventType.

FRONTAL	
SIDE	
REAR	
ROLLOVER	
NOT_SUPPORTED	
FAULT	

11.8.4 Sequence Diagrams

11.8.4.1 GetVehicleData



11.8.5 JSON Messages Examples

11.8.5.1 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,
  }
}

```

11.8.5.2 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" : NORMAL,
            "rightFront" : NORMAL,

```

```

        "leftRear" : LOW,
        "rightRear" : 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,
        "battleLevelStatus" :
THREE_LEVEL_BARS,
        "stereoAudioOutputMuted" :
: true,
        "monoAudioOutputMuted" :
false,
        "signalLevelStatus" :
NOT_PROVIDED,
        "primary AudioSource" :
MOBILE_APP,
        "eCallEventActive" :
false
    ],
    "wiperStatus" : OFF,
    "headLampStatus" :
    [
        "lowBeamsOn" : true,
        "highBeamsOn" : false
    ],
    "accPedalPosition" : 80,
    "code" : 0,
    "method" :
"VehicleInfo.GetVehicleData"
}
}
```

11.8.5.3 Error message

{

```

    "id" : 139,
    "jsonrpc" : "2.0",
    "error" :
    {
        "code" : 9,
        "message" : "The requested data is
not available",
        "data" :
        {
            "method" :
"VehicleInfo.GetVehicleData"
        }
    }
}

```

11.9 OnVehicleData

11.9.1 Description

Type:	Notification
Sender:	HMI
Purpose:	Inform about changes of vehicle data.

Initially SDL sends SubscribeVehicleData for getting the periodic updates from HMI whenever each of subscribed data types changes. OnVehicleData is expected to bring such updated values to SDL. For more details see also [11.6 SubscribeVehicleData](#) and [11.7 UnSubscribeVehicleData](#).

Note: SDL ignores all invalid notifications which come from HMI (invalid JSON, invalid data types/bounds etc)

SDL Note: SDL manages application subscriptions internally and redistributes data received via OnVehicleData between the subscribed applications by itself.

11.9.1.1 Parameters

Param Name	Type	Mandatory	Additional	Description
gps	Common.GPSData	false	-	Information related to GPS data: number of satellites, compass direction, longitude, latitude, etc. For more details please see GPSData
speed	Float	false	minvalue = 0 maxvalue = 700	The vehicle speed in kilometers per hour.
rpm	Integer	false	minvalue = 0 maxvalue = 20000	The number of revolutions per minute of the engine.

Param Name	Type	Mandatory	Additional	Description
fuelLevel	Float	false	minvalue = -6 maxvalue = 106	The fuel level in the tank (percentage).
fuelLevel_State	Common.ComponentVolumeStatus	false	-	The fuel level state should be returned: normal, low, etc. For more details please see ComponentVolumeStatus
instantFuelConsumption	Float	false	minvalue = 0 maxvalue = 25575	The instantaneous fuel consumption in microliters.
externalTemperature	Float	false	minvalue = -40 maxvalue = 100	The external temperature in degrees Celsius.
vin	String	false	maxlength = 17	Vehicle identification number.
prndl	Common.PRNDL	false	-	The position of change-speed lever: first, second, etc. For more details please see PRNDL.
tirePressure	Common.TireStatus	false	-	The following information should be returned: - The warning status (on, off, etc.) - The status of the tire pressure itself: normal, low, etc. For more details please see TireStatus.
odometer	Integer	false	minvalue = 0 maxvalue = 17000000	Odometer in km
beltStatus	Common.BeltStatus	false	-	The status of the seat belts: deployed, buckled. For more details please see BeltStatus
bodyInformation	Common.BodyInformation	false	-	The body information should be provided: park brake status, ignition and ignition stable status. For more details please see BodyInformation
deviceStatus	Common.DeviceStatus	false	-	The information on device status should be provided including signal and battery strength. For more details please see DeviceStatus.

Param Name	Type	Mandatory	Additional	Description
driverBraking	Common.VehicleDataEventStatus	false	-	The status of the brake pedal should be provided: on, off, etc. For more details please see VehicleDataEventStatus
wiperStatus	Common.WiperStatus	false	-	The status of the wipers should be provided: if they are manually on, manually off, stalled, etc. For more details please see WiperStatus
headLampStatus	Common.HeadLampStatus	false	-	Status of the head lamps should be provided: if they are on, off, etc. For more details please see HeadLampStatus
engineTorque	Float	false	minvalue = -1000 maxvalue = 2000	Torque value for engine (in Nm) for non-diesel models.
accPedalPosition	Float	false	minvalue = 0 maxvalue = 100	Accelerator pedal position (percentage depressed).
steeringWheelAngle	Float	false	minvalue = -2000 maxvalue = 2000	Current angle of the steering wheel (in degrees).

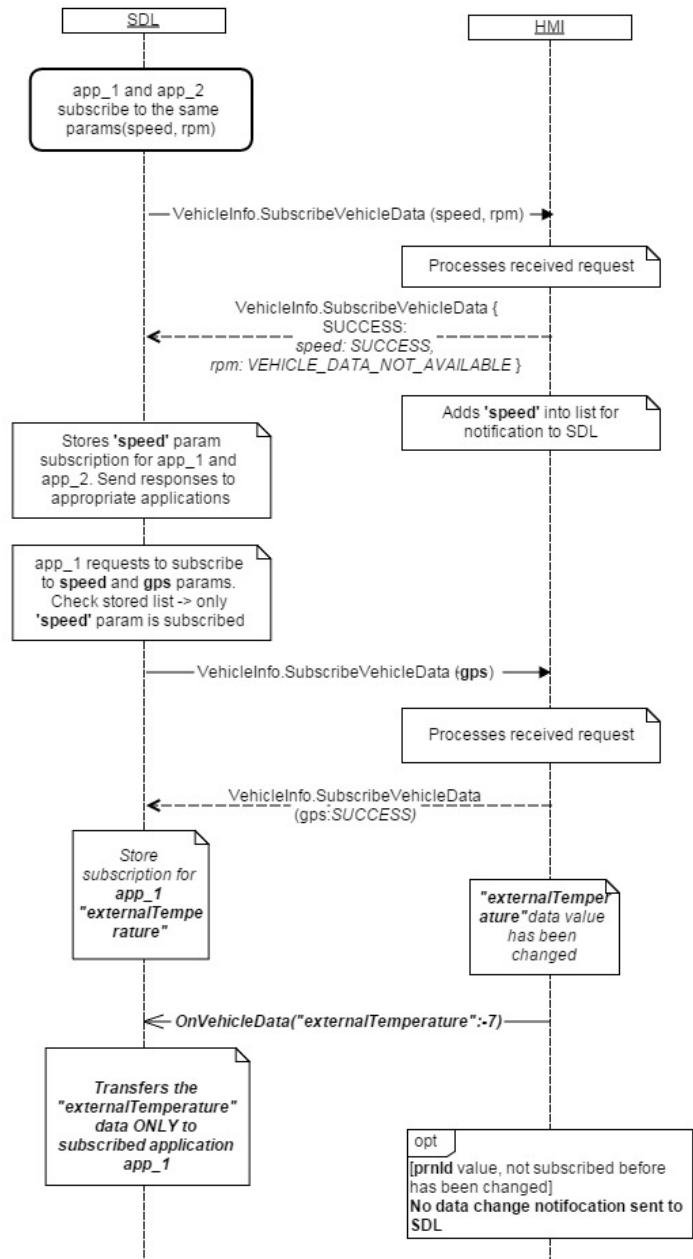
11.9.2.6 VehicleDataEventStatus

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

Element name	Value	Short Description
NO_EVENT	0	The system does not have the adequate information to send valid YES or NO states.
NO	1	The requested event is in NO state
YES	2	The requested event is in YES state.
NOT_SUPPORTED	3	The requested data is not supported

11.9.2 Sequence Diagrams

11.9.2.1 OnVehicleData



11.9.3 JSON Messages Examples

```
{
    "jsonrpc" : "2.0",
    "method" :
"VehicleInfo.OnVehicleData",
    "params" :
{
    "speed" : 60,
    "externalTemperature" : -7,
    "prndl" : THIRD,
    "odometer" : 1066,
}
```

```

        "wiperStatus" : MAN_INT_ON,
        "accPedalPosition" : 70
    }
}

```

12 Navigation Component Description

12.1 IsReady

12.1.1 Description

Type:	Function
Sender:	SDL
Purpose:	Get information if Navigation component is ready to communicate with SDL.

The request comes after HMI's readiness is confirmed via [OnReady](#) notification. SDL requires the information about whether the navigation information could be processed or provided by HMI.

Note:

If Navigation component is responded to be unavailable, SDL will not further send the requests related to it.

12.1.2 Request

12.1.2.1 Behavior

HMI must:

- 1) Check whether Navigation component is present and ready to communicate with SDL
- 2) Respond correspondingly to results of this check.

12.1.3 Response

Note:

There is a difference in message type for WebSocket and D-Bus connection (see the table below).

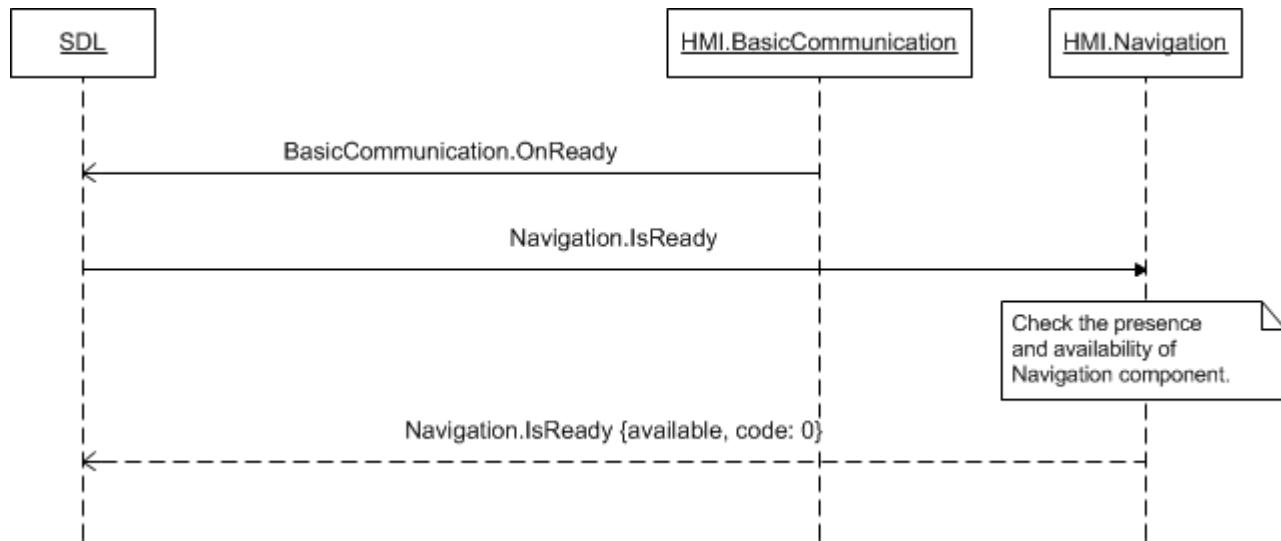
Result	Description	Message type		Message Params	Notes
		WebSocket	D-Bus		
Success	SUCCESS: HMI provides the information about Navigation component availability.	JSON response	Method return	available, code: 0	
Failure	DATA_NOT_AVAILABLE: The information about Navigation availability cannot be provided.	JSON error message	Method return	Code: 9	Applicable for this RPC result codes.
	GENERIC_ERROR: The unknown issue occurred or other codes are not applicable.			code: 22	Please see Result Enumeration for all SDL-supported codes.

12.1.3.1 Parameters

Param Name	Type	Mandatory	Description
availabe	Boolean	true	Must be - 'true' if Navigation component is present and ready to communicate - 'false' if not.

12.1.4 Sequence Diagrams

12.1.4.1 Navigation.IsReady and preceding OnReady



12.1.5 JSON Messages Examples

12.1.5.1 Request

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

12.1.5.2 Response

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

12.1.5.3 Error message

```
{
    "id" : 27,
```

```

"jsonrpc" : "2.0",
"error" :
{
  "code" : 22,
  "message" : "The unknown error has
occurred",
  "data" :
  {
    "method" :
"Navigation.IsReady"
  }
}
}

```

12.2 AlertManeuver

12.2.1 Description

Type:	Function
Sender:	SDL
Purpose:	Announce a navigation maneuver

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

12.2.2 Request

12.2.2.1 Behavior

HMI must:

1. To notify the user by TTS.Speak RPC which comes together with Navi.AlertManeuver.
2. Display the dialog of Navi.AlertManeuver with HMI-defined alert icon and one of the following:
 - Up to three soft buttons once defined within request with `softButtons` parameter
 - HMI-defined ‘Close’ soft button once the empty request has arrived (that is, without `softButtons` parameter).
 - HMI must process SystemContext behavior for AlertManeuver in the same way as for Alert

SystemContext rules:

Scenario1:

Precondition:

app1 is in FULL, app2 is in BACKGROUND, app2 is allowed to send AlertManeuver from BACKGROUND
app1 SystemContext=MAIN, app2 SystemContext=MAIN

Action:

App2 ->AlertManeuver

Expected:

AlertManeuver pop-up is shown on HMI
HMI->SDL.OnSystemContex(ALERT, app2)
HMI->SDL.OnSystemContex(OBSURED, app1)

Action:

Alert is closed

Expected:

AlertManeuver pop-up is shown on HMI
HMI->SDL.OnSystemContex(MAIN, app2)
HMI->SDL.OnSystemContex(MAIN, app1)

AppID rules for AlertManeuver when it causes

SystemContext updates:

1. AppID should not be sent for MENU and HMI_OBCSURED system contexts. Only the apps in FULL may get these updates.
- 2 .In case OnSystemContext of MENU and HMI_OBCSURED are sent with appId by HMI, appId is ignored and anyway the notification is transferred to the app in FULL HMI Level
3. OnSystemContext appId parameter should be mandatory for MAIN and ALERT values. In case there's no such application with appId received, the notifications will be ignored by SDL

SDL Note: In case HMI does not respond SDL's request during request or/SDL default timeout, SDL will send GENERIC_ERROR result code to the corresponding mobile app's request

12.1.2.1 Parameters

Param Name	Type	Mandatory	Additional	Description
softButtons	Common.SoftButton	false	Array = true minsize = 0 maxsize = 3	Up to three soft buttons defined. If omitted, only the system defined "Close" soft button should be displayed (if applicable).
appId	Integer	true		ID of the application which requested this RPC

12.2.3 Response

Note:

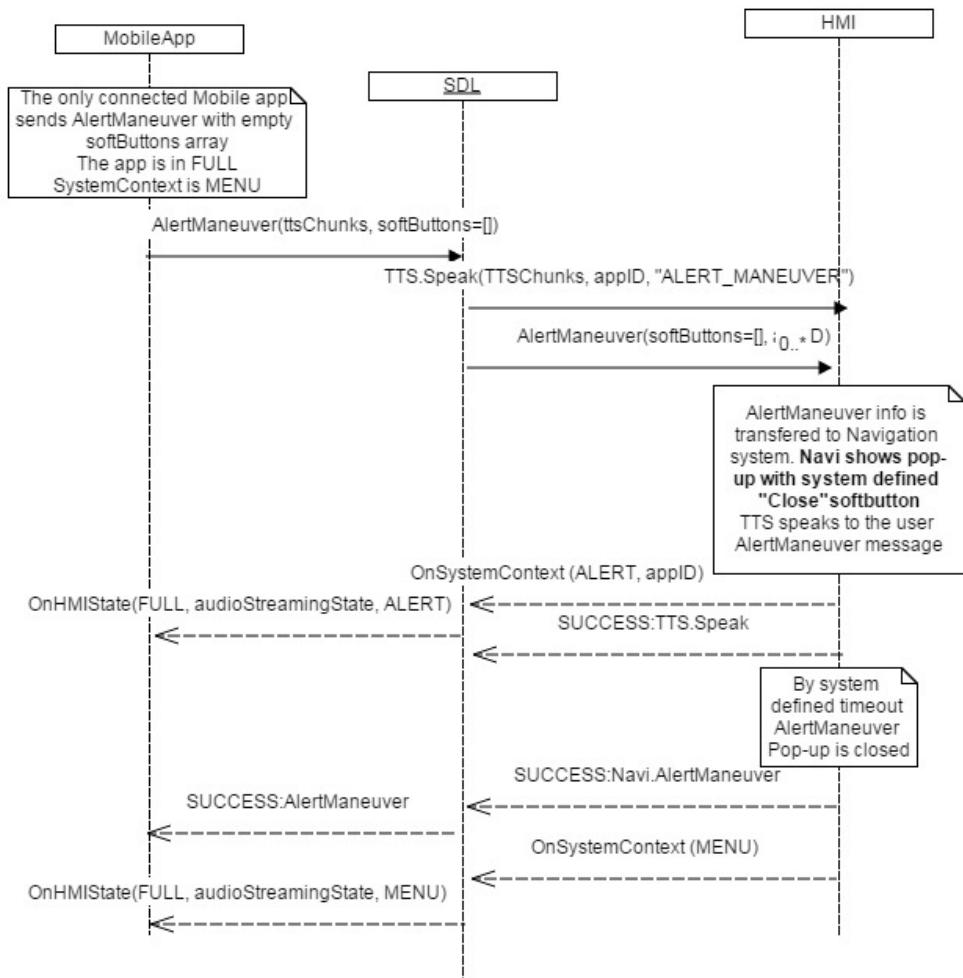
There is a difference in message type for WebSocket and D-Bus connection (see the table below).

Result	Description	Message type		Message Params	Notes
		WebSocket	D-Bus		
Success	SUCCESS: HMI displayed the requested dialog of navigation maneuver.	JSON response	Method return	code: 0	Applicable for this RPC result codes. Please see Result Enumeration for all SDL-supported codes.
Failure	INVALID_DATA: The data sent is invalid (invalid JSON syntax or out of bound parameters)	JSON error message	Method return	code: 11	The check for invalid data is actually performed by SDL. Optionally, HMI may also perform the check, In this case the code

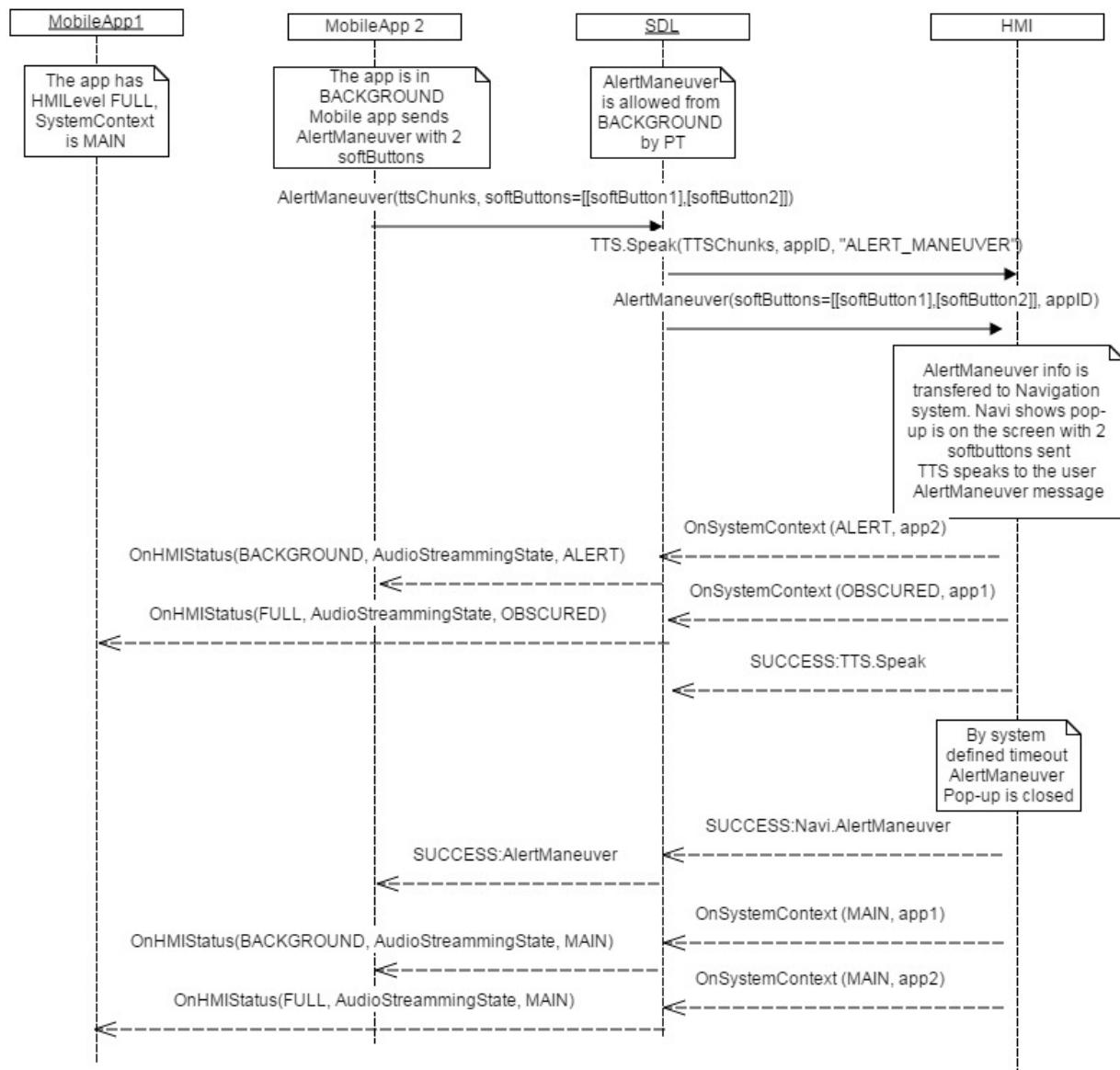
					must be returned
	ABORTED: The dialog is aborted with the higher priority RPC.			Code : 5	Applicable for this RPC result codes. Please see Result Enumeration for all SDL-supported codes.
	INVALID_ID appID is invalid			code:13	The check for invalid appID is actually performed by SDL. Optionally, HMI may also perform the check, In this case the code must be returned
	UNSUPPORTED_RESOURCE When icon is sent by SDL but HMI doesn't support the type of images sent by SDL (STATIC/DYNAMIC).			code:2	Applicable for this RPC result codes. Please see Result Enumeration for all SDL-supported codes.
	GENERIC_ERROR 1) The unknown issue occurred or other codes are not applicable.			code: 22	

12.2.4 Sequence Diagrams

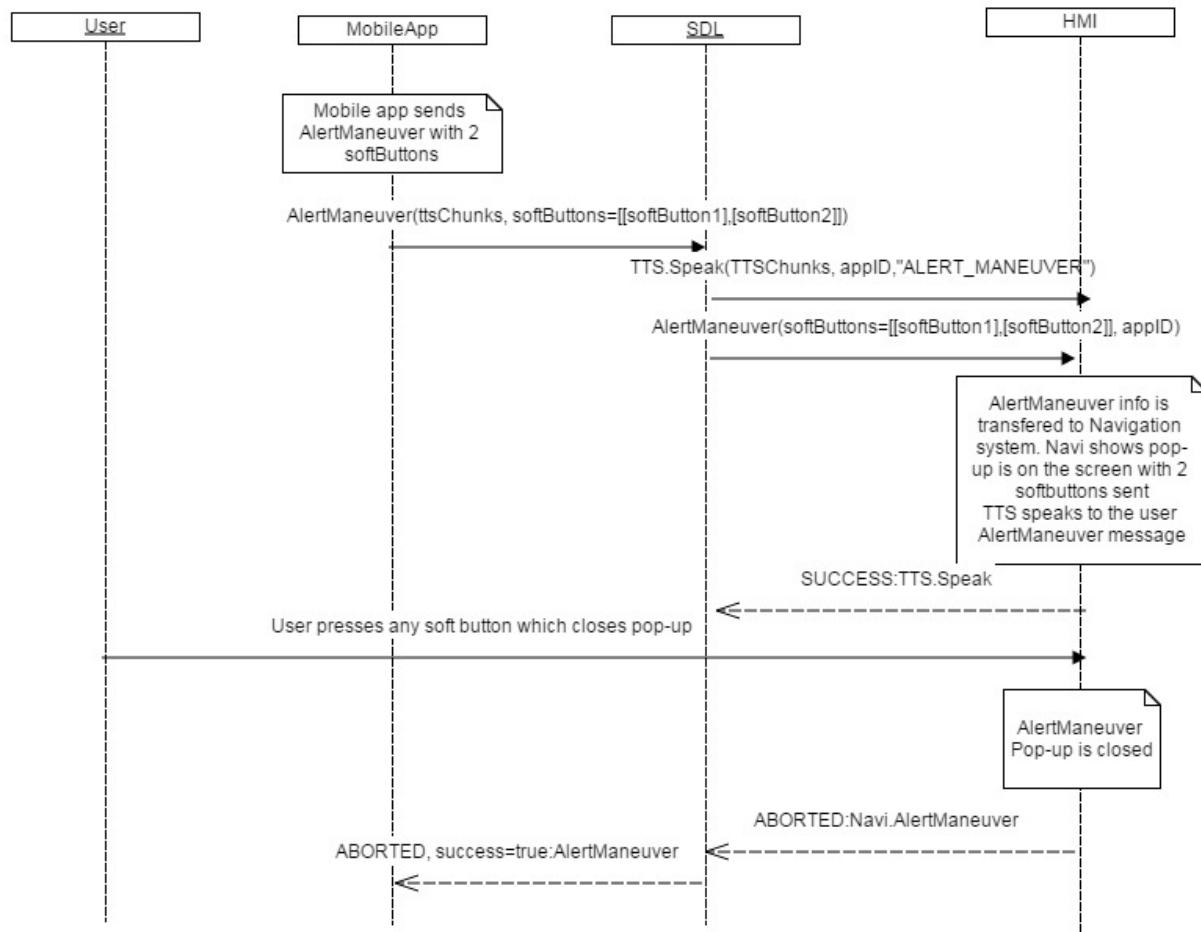
12.2.4.1 AlertManeuver Default success path



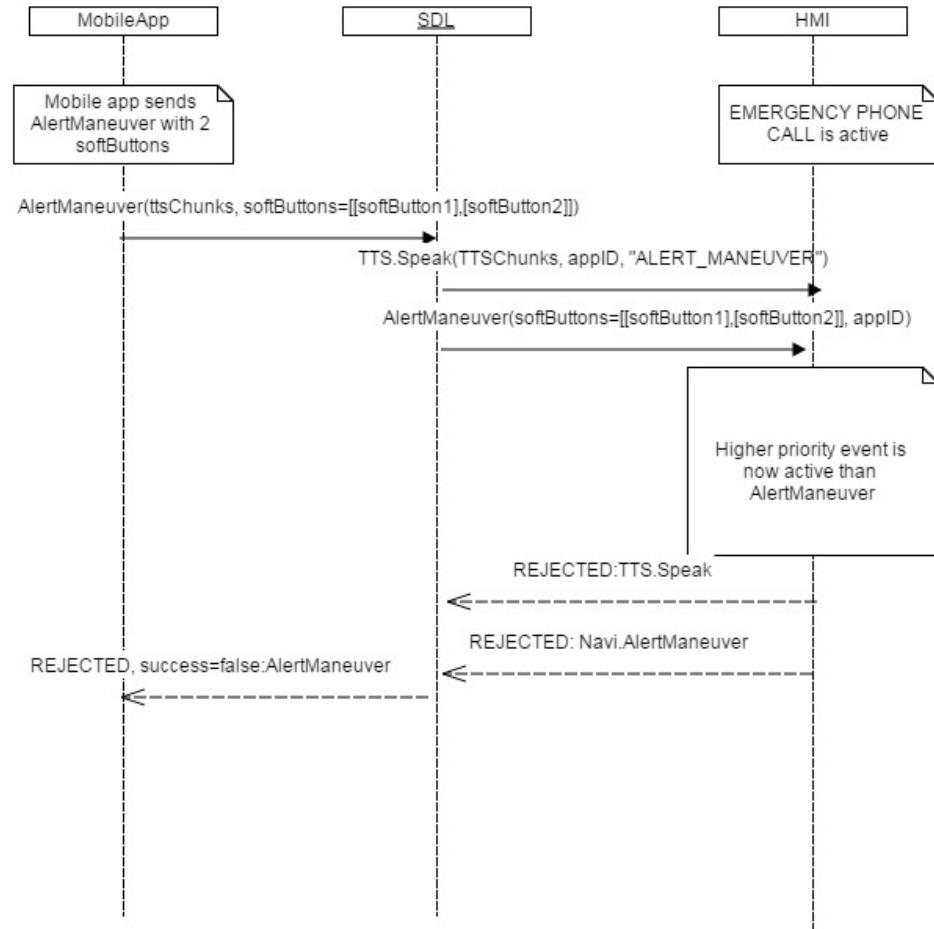
12.2.4.2 AlertManeuver from BACKGROUND success path



12.2.4.3 AlertManeuver ABORTED



12.2.4.4 AlertManeuver REJECTED



12.2.5 JSON Messages Examples

12.2.5.1 Request Navi.AlertManeuver

```
{
  "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}

```

12.2.5.2 Response Navi.AlertManeuver

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

12.2.5.3 Request TTS.Speak for AlertManeuver

```
{
    "id" : 144,
    "jsonrpc" : "2.0",
    "method" : "TTS.Speak",
    "params" :
    {
        "ttsChunks" :
        [
            {
                "text" : "Attention! Turn
Left"
            },
            "appID":96
        ]
    }
}
```

12.2.5.4 Response TTS.Speak for AlertManeuver

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

12.2.5.5 Error message

```
{
    "id" : 143,
```

```

"jsonrpc" : "2.0",
"error" :
{
    "code" : 13,
    "message" : "A command cannot be
executed because there is NO specified
with appID application
registered",
    "data" :
    {
        "method" :
"Navigation.AlertManeuver"
    }
}
}

```

12.3 ShowConstantTBT

12.3.1 Description

Type:	Function
Sender:	SDL
Purpose:	Update navigation information of the embedded navigation system

SDL request to display navigation information sent from the mobile application to the embedded navigation system. This information may reproduce the data about the current navigation state (e.g. destination place, icon of the destination place, total distance, distance to maneuver, scale, softbuttons to operate the information on the screen and other).

12.3.2 Request

12.3.2.1 Behavior **HMI must:**

- 1) Display all obtained navigation information from SDL
 - In case embedded navi screen is now not active, the data will be shown the next time the user will open embedded navigation layout
 - In case embedded navi screen is now active layout, the data obtained must be shown at ones
- 2) Send a response on a request

SDL Note: In case HMI does not respond SDL's request during SDL default timeout (10 sec), SDL will return GENERIC_ERROR code to the corresponding mobile app's request

12.3.2.1 Parameters

Param Name	Type	Mandatory	Additional	Description
------------	------	-----------	------------	-------------

Param Name	Type	Mandatory	Additional	Description
navigationTexts	Common.TextFieldStruct	true	Array = true minsize = 0 maxsize = 5	The text to be displayed in up to four fields on Navigation display: - navigationText1 - navigationText2 - ETA - totalDistance For more details please see TextFieldStruct, section 14.2.11
turnIcon	Common.Image	false	-	The icon to be displayed. For more details please see Image, section 14.2.14 .
nextTurnIcon	Common.Image	false	-	
distanceToManeuver	Float	true	minvalue = 0 maxvalue = 1000000000	The distance from the previous maneuver till the next one. HMI may use these data for calculating the progress bar.
distanceToManeuverScale	Float	true	minvalue = 0 maxvalue = 1000000000	Fraction of distance from the AlertManeuver triggering till the next maneuver. HMI may use these data for calculating the progress bar.
maneuverComplete	Boolean	false	-	If 'true', the maneuver has been completed and the AlertManeuver overlay must be cleared. If omitted the value must be assumed as 'false'.
softButtons	Common.SoftButton	false	Array = true minsize = 0 maxsize = 3	SDL may request to draw up to three soft buttons for Navigation. If the empty array is sent or the parameter is omitted, the currently displayed soft button values must remain unchanged. For more details please see SoftButton, section 14.2.13
appID	Integer	true	-	ID of the application that concerns this RPC.

12.3.3 Response

Note:

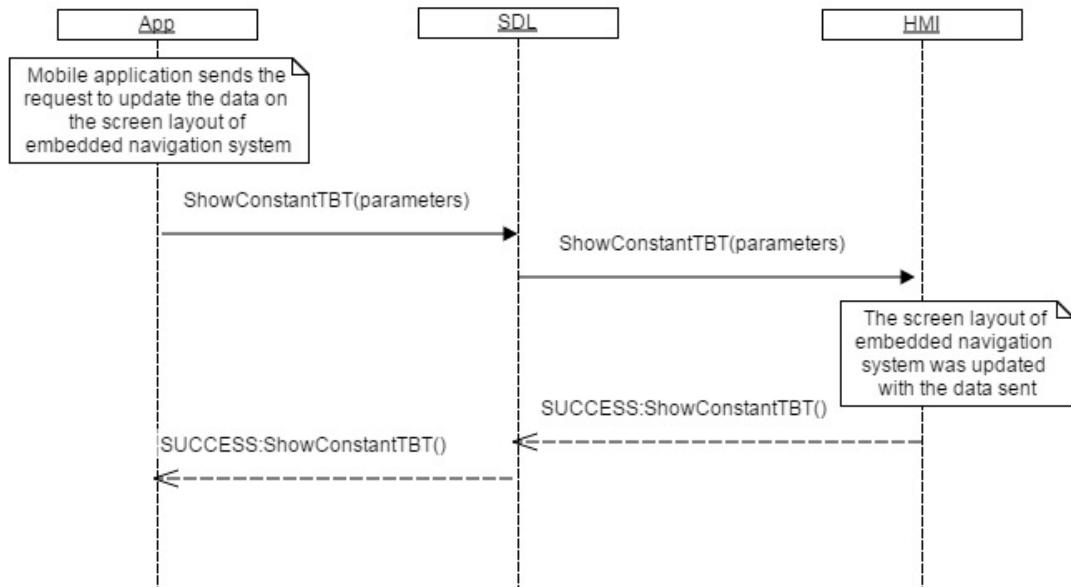
There is a difference in message type for WebSocket and D-Bus connection (see the table below).

Result	Description	Message type		Message Params	Notes
		WebSocket	D-Bus		
Success	SUCCESS: HMI executed the request successfully.	JSON response	Method return	code: 0	

	INVALID_DATA: The data sent is invalid (invalid JSON syntax, parameters out of bounds or of wrong type)	JSON error message	Method return	code: 11	The check for invalid data is actually performed by SDL. Optionally, HMI may also perform the check, In this case the code must be returned
Failure	INVALID_ID appId is invalid (e.g. the app with current id doesn't exist)			code: 13	The check for appId is actually performed by SDL. Optionally, HMI may also perform the check, In this case the code must be returned
	UNsupported_RESOURCE When icon is sent by SDL but HMI doesn't support the type of images sent by SDL (STATIC/DYNAMIC).			code: 2	
	REJECTED: HMI is expected to return ABORTED result code in case HMI is currently busy with a higher-priority event.			Code: 45	
	GENERIC_ERROR 1) The unknown issue occurred or other codes are not applicable.			code: 22	

12.3.4 Sequence Diagrams

12.3.4.1 ShowConstantTBT



12.3.5 JSON Messages Examples

12.3.5.1 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  
            }  
    }  
}
```

12.3.5.2 Response

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

```

    }
}
```

12.3.5.3 Error message

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

12.4 UpdateTurnList

12.4.1 Description

Type:	Function
Sender:	SDL
Purpose:	Update turn list of embedded navigation layout

SDL sends UpdateTurnList to notify the embedded navigation system about the next navigation maneuvers which will take place on a chosen route.

12.4.2 Request

12.4.2.1 Behavior

HMI must:

1) Display an updates of the TurnList received from SDL and one of the following:

- One soft buttons defined within request with `softButtons` parameter
- HMI-defined ‘Back’ button once the empty request has arrived (that is, without `softButtons` parameter).

SDL Note: In case HMI does not respond SDL's request during SDL default timeout (10 sec), SDL will return `GENERIC_ERROR` code to the corresponding mobile app's request

12.4.2.1 Parameters

Param Name	Type	Mandatory	Additional	Description
turnList	Common.Turn	false	Array = true minsize = 1 maxsize = 100	The array of objects that consist of: - text string to be displayed in navigation field - image for turn prompt.

Param Name	Type	Mandatory	Additional	Description
				For more details please see Turn, section
softButtons	Common.SoftButton	false	Array = true minsize = 0 maxsize = 1	One soft button defined. For more details regarding the type/highlighted or not/etc. please see SoftButton, section

12.4.2.2 Turn

The structure represents the information for TBT navigation Turn.

Param Name	Type	Mandatory	Description
navigationText	Common.TextFieldStruct	false	Contains the information text and the field for the text to be displayed in. Uses navigationText from TextFieldName
turnIcon	Common.Image	false	The image that represents the information about the turn. See Image

12.4.3 Response

Note:

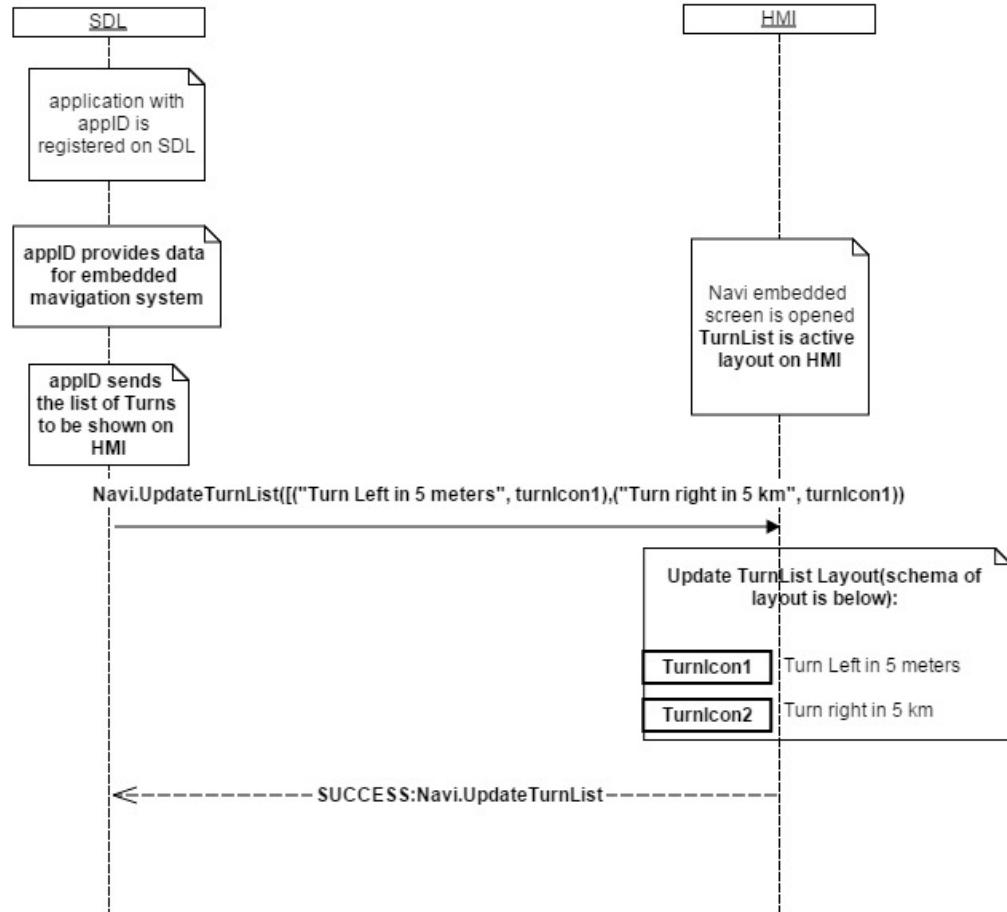
There is a difference in message type for WebSocket and D-Bus connection (see the table below).

Result	Description	Message type		Message Params	Notes
		WebSocket	D-Bus		
Success	SUCCESS: HMI executed the RPC successfully.	JSON response	Method return	code: 0	
Failure	INVALID_DATA: The data sent is invalid (invalid JSON syntax, parameters out of bounds or of wrong type)	JSON error message	Method return	code: 11	Applicable for this RPC result codes.
	INVALID_ID appId is invalid (e.g. the app with current id doesn't exist)			code: 13	
	UNSUPPORTED_RESOURCE When icon is sent by SDL but HMI doesn't support the type of images sent by SDL (STATIC/DYNAMIC).			code: 2	Please see Result Enumeration for all SDL-supported codes.
	GENERIC_ERROR 1) The unknown issue occurred or other codes are not applicable.			code: 22	



12.4.4 Sequence Diagrams

12.4.4.1 UpdateTurnList



12.4.5 JSON Messages Examples

12.4.5.1 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
    ]
}
}
```

12.4.5.2 Response

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

12.4.5.3 Error message

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

12.5 StartStream

12.5.1 Description

Type:	Notification
Sender:	SDL
Purpose:	To initiate the channel between SDL and HMI to stream video data further

When an application opens the 11th service, SDL sends StartStream to HMI. It initiates the channel between SDL and HMI for streaming video raw data which is being sent by navigation application. The request just initiates the connection by URL defined in a request, but not starts read the data. [OnVideoDataStreaming](#) notification is responsible for notifying to start/end read the data from the channel.

Note:

- Video streaming is allowed by SDL only for the apps in FULL and LIMITED HMI levels

HMI must:

- prepare to read the raw video data from the URL defined in StartStream request

12.5.2 Request**12.5.2.1 Parameters**

Param Name	Type	Mandatory	Additional	Description
url	String	true	Minlength = 21 maxlength = 500	URL that HMI will start playing after OnVideoDataStream(available:true)
appID	Integer	true	-	ID of the application requested this RPC

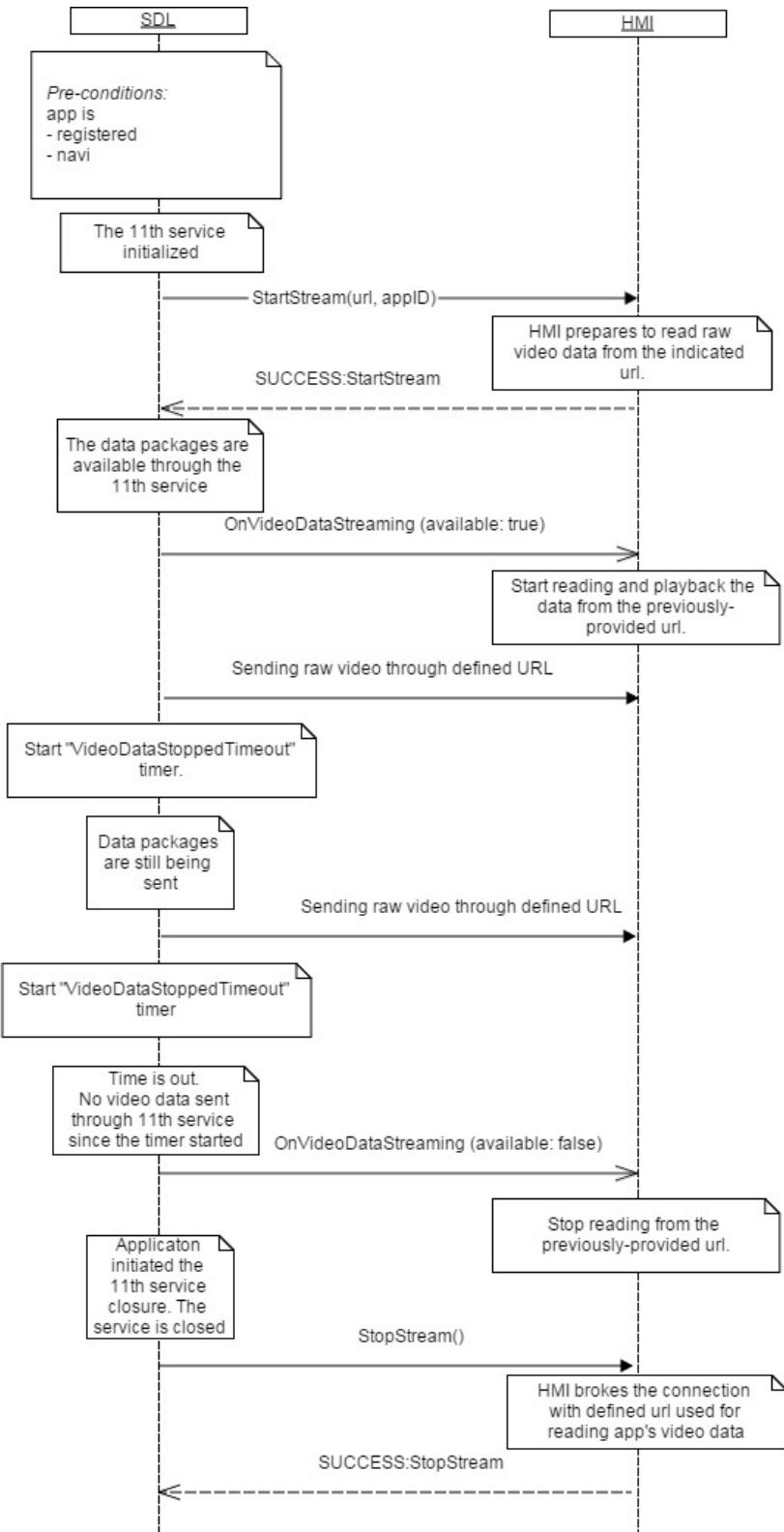
12.5.3 Response**12.5.3.1 Parameters****Note:**

There is a difference in message type for WebSocket and D-Bus connection (see the table below).

Result	Description	Message type		Message Params	Notes
		WebSocket	D-Bus		
Success	SUCCESS: HMI executed the RPC successfully.	JSON response	Method return	code: 0	
Failure	GENERIC_ERROR: The unknown issue occurred or other codes are not applicable.			code: 22	

12.5.4 Sequence Diagrams

12.5.4.1 StartStream



12.5.5 JSON Messages Examples

12.5.5.1 Request

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

12.5.5.2 Response

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

12.5.5.3 Error message

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

12.6 StopStream

12.6.1 Description

Type:	Notification
Sender:	SDL
Purpose:	To initiate the video streaming channel closure between SDL and HMI

When an application closes the 11th service, SDL sends StopStream to HMI. It initiates the channel closure between SDL and HMI which is used for streaming video raw data sent by navigation application. The request just interrupts the connection defined in StartStream request.

Note:

- Video streaming is allowed by SDL only in FULL and LIMITED HMI levels

HMI must:

- close the channel used for reading the raw video data from the URL defined in StartStream request

12.6.2 Request

12.6.2.1 Parameters

Param Name	Type	Mandatory	Additional	Description
appID	Integer	true	-	ID of the application requested this RPC

12.6.3 Response

12.6.3.1 Parameters

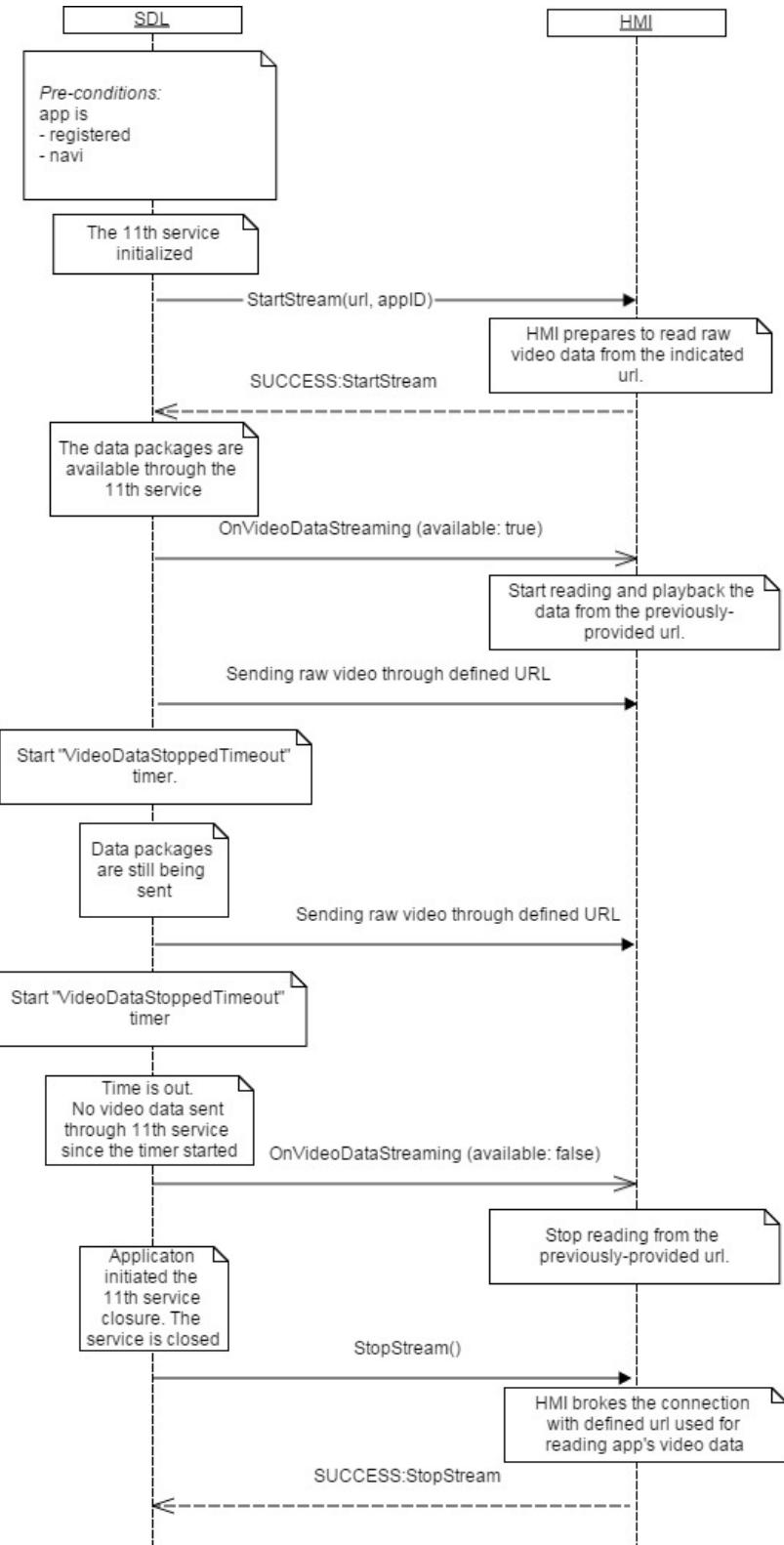
Note:

There is a difference in message type for WebSocket and D-Bus connection (see the table below).

Result	Description	Message type		Message Params	Notes
		WebSocket	D-Bus		
Success	SUCCESS: HMI executed the RPC successfully.	JSON response	Method return	code: 0	
Failure	GENERIC_ERROR: The unknown issue occurred or other codes are not applicable.			code: 22	

12.6.4 Sequence Diagrams

12.6.2.1 StopStream



12.6.5 JSON Messages Examples

12.6.5.1 Request

```
{
    "jsonrpc" : "2.0",
    "method" : "video.startStream",
    "params" : {
        "url": "http://192.168.1.100:8080/video.mp4",
        "appId": "com.example.app"
    }
}
```

```

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

```

12.6.5.2 Response

```

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

```

12.6.5.3 Error message

```

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

```

12.7 StartAudioStream

12.7.1 Description

Type:	Notification
Sender:	SDL
Purpose:	To initiate the channel between SDL and HMI to stream audio data further

The request initiate the channel between SDL and HMI for streaming audio raw data which is being sent by navigation application. The request just initiates the connection by URL defined in a request, but not starts read the data. [OnAudioDataStream](#) notification is responsible for notifying to start/end read the data from the channel.

Note:

- audio streaming is allowed by SDL only in FULL and LIMITED HMI levels

HMI must:

- create a channel for getting raw audio data streaming via defined URL

12.7.2 Request

12.7.1.1 Parameters

Param Name	Type	Mandatory	Additional	Description
url	String	true	Minlength = 21 maxlength = 500	URL that HMI gets data from .
appID	Integer	true	-	ID of the application requested this RPC

12.7.3 Response

12.7.3.1 Parameters

Note:

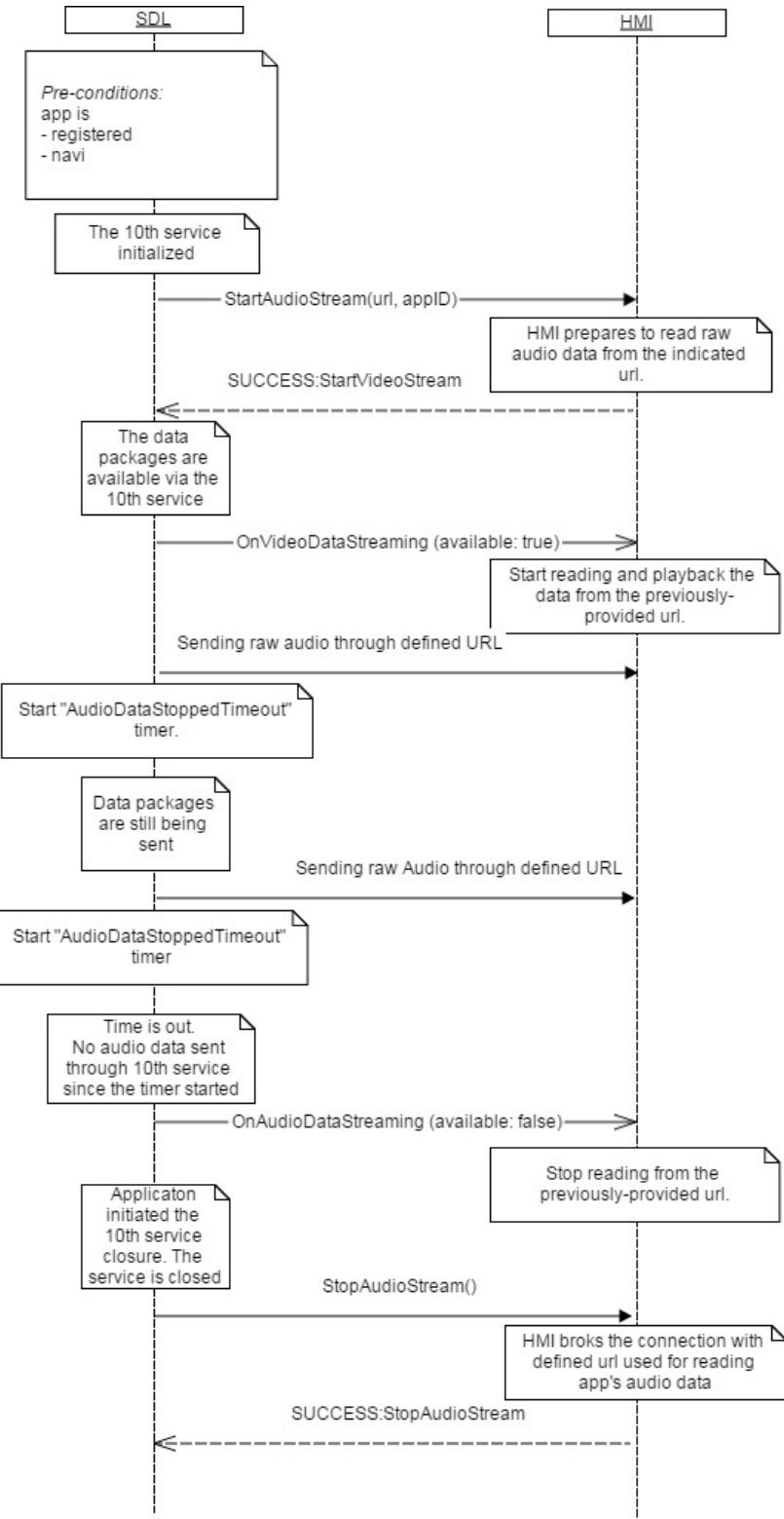
There is a difference in message type for WebSocket and D-Bus connection (see the table below).

Result	Description	Message type		Message Params	Notes
		WebSocket	D-Bus		
Success	SUCCESS: HMI executed the RPC successfully.	JSON response	Method return	code: 0	
Failure	GENERIC_ERROR: The unknown issue occurred or other codes are not applicable.			code: 22	

12.7.4 Sequence Diagrams

12.5.4.1

StartAudioStream/StopAudioStream/OnAudioDataStreaming



12.7.5 JSON Messages Examples

12.7.5.1 Request

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

12.7.5.2 Response

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

12.7.5.3 Error message

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

12.8 StopAudioStream

12.8.1 Description

Type:	Notification
Sender:	SDL
Purpose:	To initiate the audio streaming channel closure between SDL and HMI.

When an application closes the 10th service, SDL sends StopAudioStream to HMI. It initiates the channel closure between SDL and HMI which was used for streaming audio raw data sent by navigation application. The request just closes the connection defined in StartAudioStream request.

Note:

- Audio streaming is allowed by SDL only in FULL and LIMITED HMI levels

HMI must:

- close the channel used for reading the raw audio data through the URL defined in StartAudioStream request

12.8.2 Request

12.8.2.1 Parameters

Param Name	Type	Mandatory	Additional	Description
appID	Integer	true	-	ID of the application requested this RPC

12.8.3 Response

12.8.3.1 Parameters

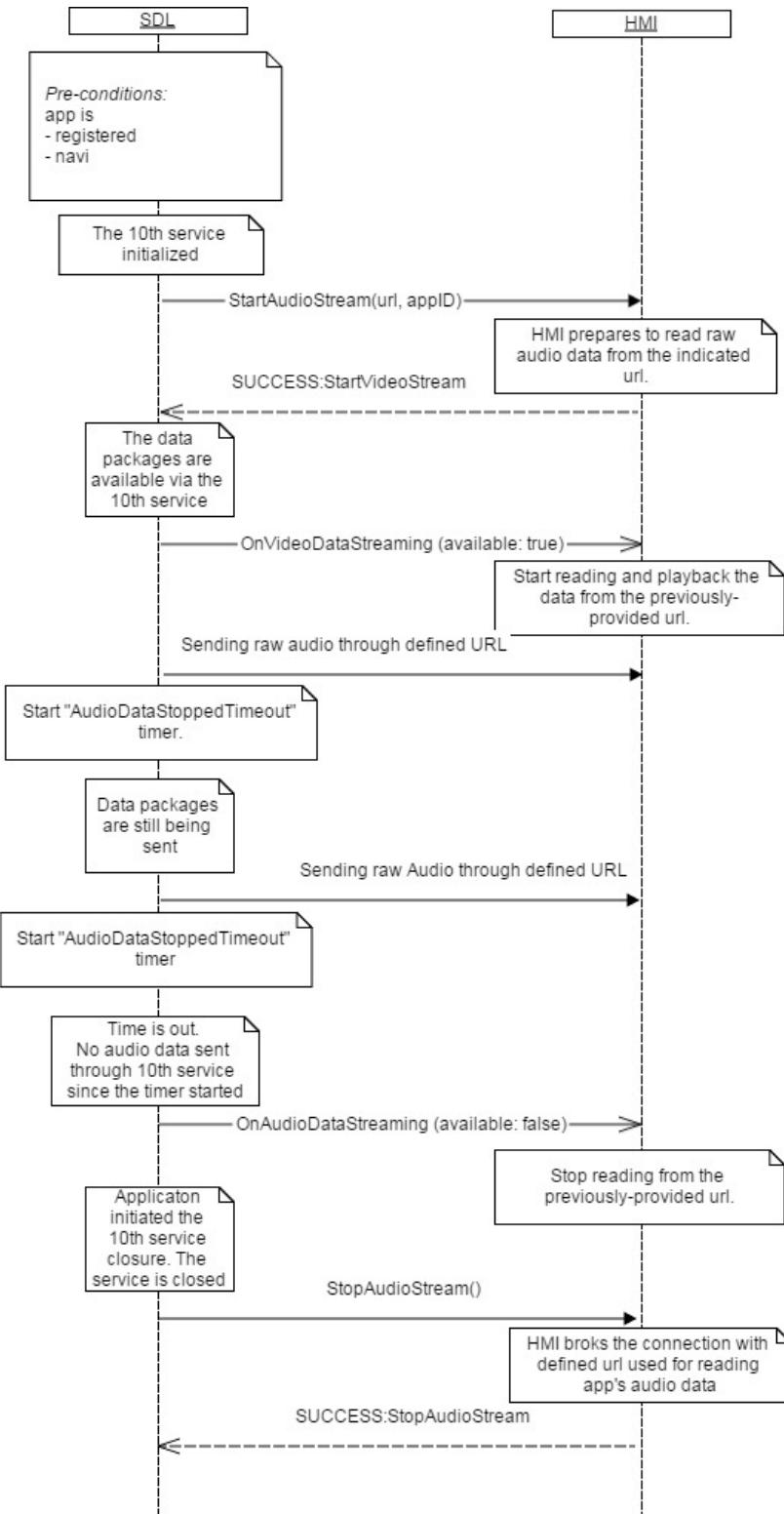
Note:

There is a difference in message type for WebSocket and D-Bus connection (see the table below).

Result	Description	Message type		Message Params	Notes
		WebSocket	D-Bus		
Success	SUCCESS: HMI executed the RPC successfully.	JSON response	Method return	code: 0	
Failure	GENERIC_ERROR: The unknown issue occurred or other codes are not applicable.			code: 22	

12.8.4 Sequence Diagrams

12.8.4.1 StopAudioStream



12.8.5 JSON Messages Examples

12.8.5.1 Request

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

12.8.5.2 Response

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

12.8.5.3 Error message

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

12.9 OnTBTClientState

12.9.1 Description

Type:	Notification
Sender:	HMI
Purpose:	Provide the information about TBT Client state

HMI must:

- 1) HMI must provide SDL with notifications specific to the current Turn-By-Turn client status on the module

OnTBTClientState informs SDL about some event or state happened on HMI Turn-By-Turn client. Further, SDL transfers this data to the application which uses this information for the navigation actual data support.
Application may provide some information back to HMI as a result of OnTBTClientState receival.

Note: SDL ignores all invalid notifications which come from HMI (invalid JSON, invalid data types/bounds etc)

12.9.1.1 Parameters

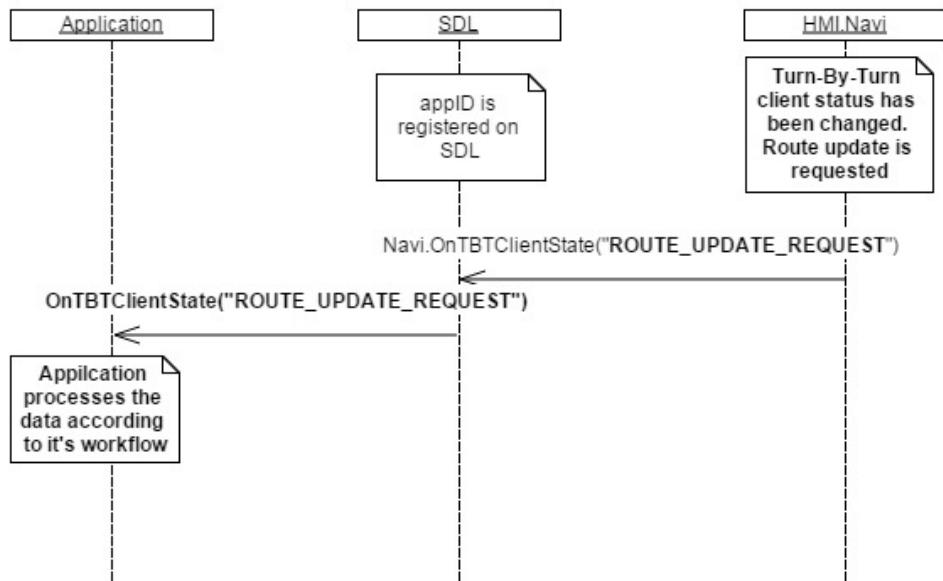
Param Name	Type	Mandatory	Description
state	Common.TBTState	true	Current State of TBT client. For more details please see TBTState

12.9.1.2 TBTState

Element name	Short Description
ROUTE_UPDATE_REQUEST	HMI requested for the route update.
ROUTE_ACCEPTED	Confirmation about accepting the route.
ROUTE_REFUSED	The current route has been refused.
ROUTE_CANCELLED	Cancelling the route.
ETA_REQUEST	HMI requested to estimate the time of arrival
NEXT_TURN_REQUEST	Navigation requested the information about the next turn.
ROUTE_STATUS_REQUEST	Navigation requested the information about the route status
ROUTE_SUMMARY_REQUEST	Navigation requested the information about the route summary
TRIP_STATUS_REQUEST	Navigation requested the information about the trip status.
ROUTE_UPDATE_REQUEST_TIMEOUT	Navigation requested the route update request by timeout

12.9.2 Sequence Diagrams

12.9.2.1 OnTBTClientState ROUTE_UPDATE_REQUEST



12.9.3 JSON Messages Examples

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

12.10 SendLocation

12.10.1 Description

Type:	Function
Sender:	SDL
Purpose:	To allow an application to send a destination to the embedded navigation system.

12.10.2 Request

12.10.2.1 Behavior

HMI must:

- 1) Receive the data from SDL and transfer it to navigation embedded module. Navigation embedded HU system should update the location data on the embedded navi screen. Initially, this data is transferred from mobile application to notify the embedded navi system about the location coordinates where the user navigates to.
- 2) Send a SUCCESS response to SDL in case the destination data has been obtained by HMI

SDL Note: In case HMI does not respond SDL's request during SDL default timeout (10 sec), SDL will return GENERIC_ERROR code to the corresponding mobile app's request

12.10.2.2 Parameters

Param Name	Type	Mandatory	Additional	Description
longitudeDegrees	Float	true	Minvalue = -180 Maxvalue = 180	
latitudeDegrees	Float	true	Minvalue = -90 Maxvalue = 90	
locationName	String	false	Maxlength = 500	
addressLines	String	false	Maxlength = 500 minsize= 0 maxsize= 4 array= true	
phoneNumber	String	false	Maxlength = 500	
locationImage	Common.Image	false		

12.10.3 Response

Note:

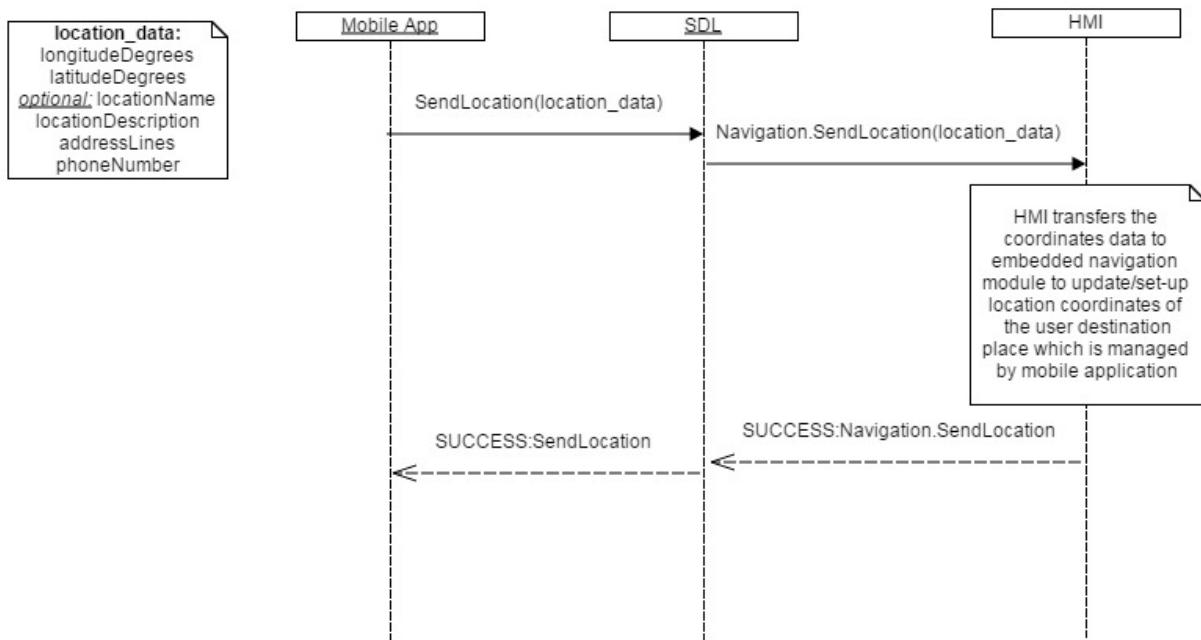
There is a difference in message type for WebSocket and D-Bus connection (see the table below).

Result	Description	Message type		Message Params	Notes
		WebSocket	D-Bus		
Success	SUCCESS:	JSON response	Method return	code: 0	
	UNSUPPORTED_RESOURCE 1) When icon is sent by SDL but HMI doesn't support the type of images sent by SDL (STATIC/DYNAMIC). 2) In case UI is not supported (UI.IsReady returned false)	JSON response	Method return	code: 2	

	<p>WARNINGS</p> <p>1) In case the requested image to display is corrupted or does not exist by the defined path 2) Some of the parameters sent by SDL aren't supported as text fields on HMI. Supported values should be displayed/processed as expected</p>			<p>HMI displays all other requested info except of missed images and returns WARNINGS code with problem description</p> <p>Note: from SDL's point of view the request is executed successfully, so mobile app will get WARNINGS response, but value success=true</p>
	<p>INVALID_DATA: The data sent is invalid (invalid JSON syntax or parameters out of bounds or of wrong type)</p>		<p>code: 21</p>	
	<p>REJECTED HMI rejects RPCs in terms of HMI-matrix (higher priority event is now active)</p>		<p>code: 11</p>	
	<p>GENERIC_ERROR</p> <p>1) The unknown issue occurred or other codes are not applicable.</p>		<p>code: 4</p>	
			<p>code: 22</p>	

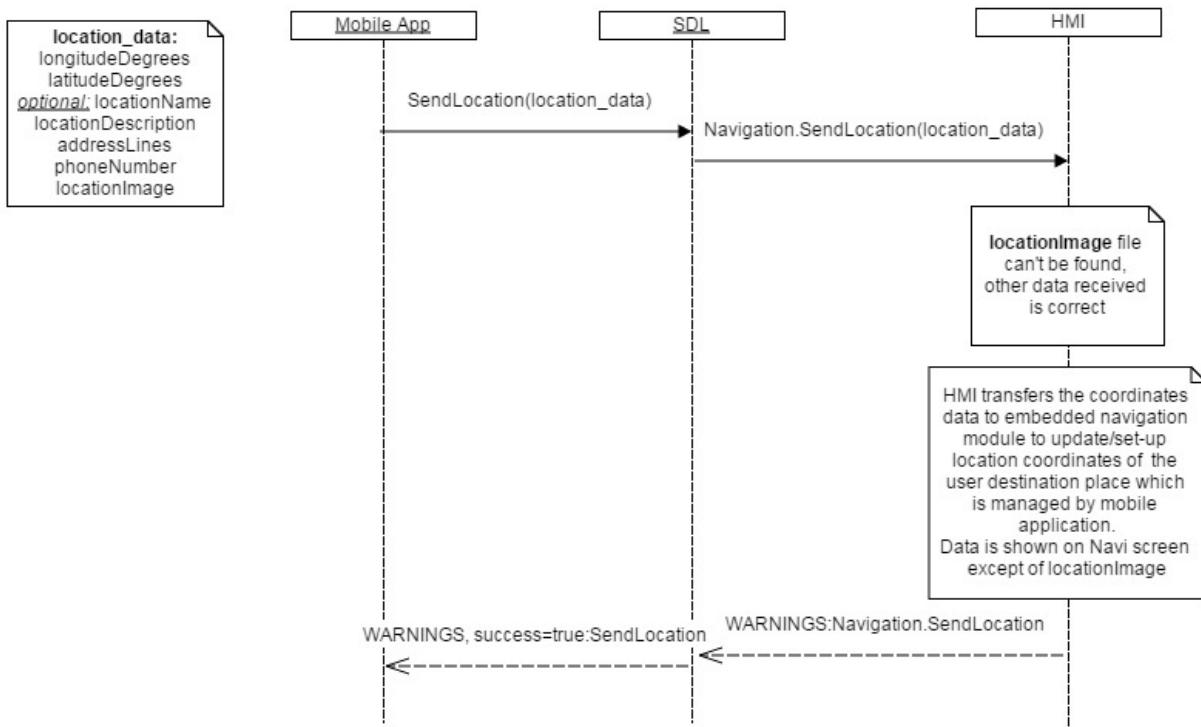
12.10.4 Sequence Diagrams

12.10.4.1 SendLocation Success scenario

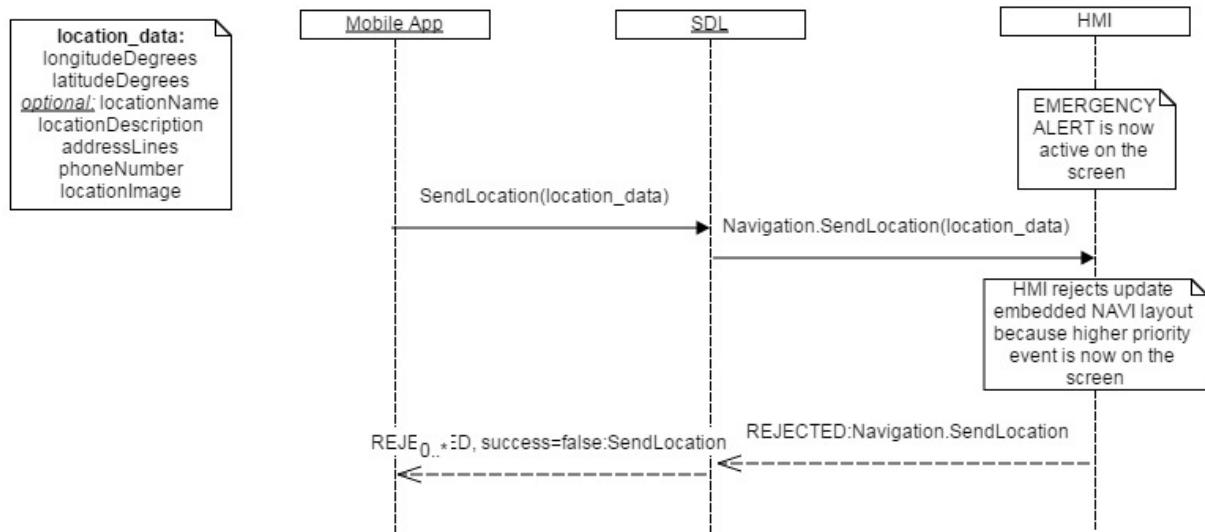


12.10.4.2 SendLocation Failure scenario

WARNINGS



12.10.4.3 SendLocation Failure scenario REJECTED



12.10.5 JSON Messages Examples

12.10.5.1 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
        }
    }
}
```

12.10.5.2 Response

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

12.10.5.3 Error message

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

12.11 OnAudioDataStreaming

12.11.1 Description

Type:	Notification
Sender:	SDL
Purpose:	To notify about raw audio data sending over the URL provided via StartAudioStream SDL's request.

Note:

- Audio streaming is allowed by SDL only in FULL and LIMITED HMI levels

HMI must:

- 1) get the audio through the URL
- 2) playback the received audio data via audio system

12.11.1.1 Parameters

Param Name	Type	Mandatory	Description
available	Boolean	true	If "true" - audio data started. If "false" - audio data stopped.

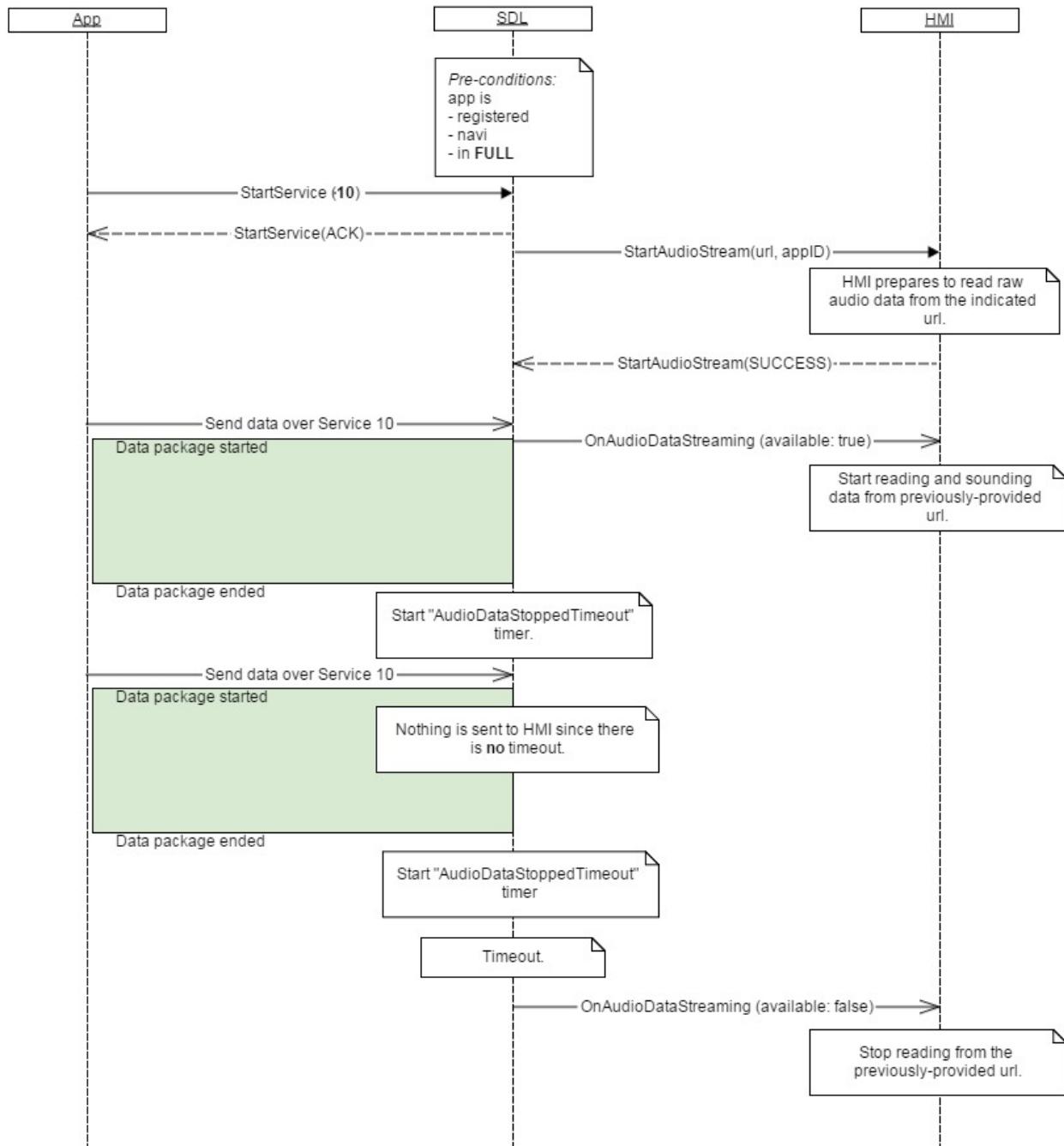
12.11.2 Sequence Diagrams

12.11.2.1 OnAudioDataStreaming SDL<->HMI



12.11.2.1 OnAudioDataStreaming full diagram

App<->HMI



12.11.3 JSON Messages Examples

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

12.12 OnVideoDataStreaming

12.12.1 Description

Type:	Notification
Sender:	SDL->HMI
Purpose:	To notify about raw video data sending over the URL provided via StartStream SDL's request.

Note:

- Video streaming is allowed by SDL only in FULL and LIMITED HMI levels

HMI must:

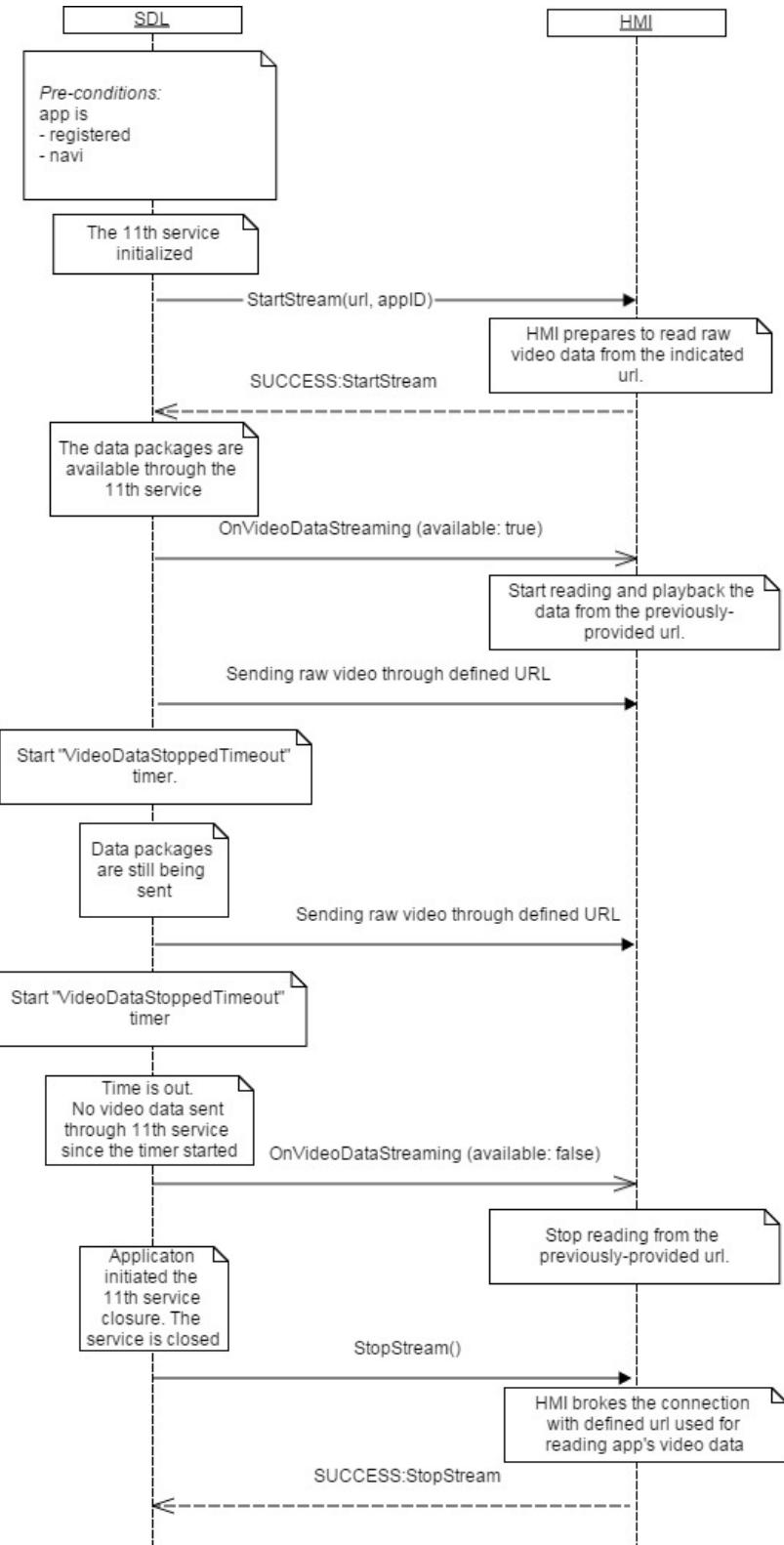
- 1) get the audio through the URL defined in [StartStream](#)
- 2) playback the received video data

12.12.1.1 Parameters

Param Name	Type	Mandatory	Description
available	Boolean	true	If "true" - video data is started to be sending. If "false" - video data sending has been stopped.

12.12.2 Sequence Diagrams

12.12.2.1 OnVideoDataStreaming



12.12.3 JSON Messages Examples

```
{
  "jsonrpc": "2.0",
  "method": "Navigation.OnVideoDataStreaming",
```

```

"params" :
{
  "available" : true
}

```

13 SDL Component Description

Notes about ‘SDL’ component:

A. ‘SDL’ names a component that is a part of SDL. HMI has the ability to request and receive definite information from SDL using ‘SDL’ interface, in contrast to communicating using other interfaces of HMI_API.

B. HMI-SDL communication with ‘SDL’ component over WebSocket (WS):

B.1. Both parties should use the WS registered with the name of ‘BasicCommunication’ (see [section 2.1 Connection Opening of chapter 2](#) [WebSocketTransport](#)). Separate WS connection should not be opened for communication with ‘SDL’ component.

B.2. Both parties should send JSON messages with the field of:

"method" : "SDL.<RPC_name>" (for example,
"SDL.ActivateApp")

when communicating using ‘SDL’ ‘interface’.

13.1 SDL.ActivateApp

13.1.1 Description

Type:	Function
Sender:	HMI
Purpose:	Inform about User has activated the application and get the permissions for it.

SDL needs SDL.ActivateApp request to:

1. Know that the named application has been chosen by the User.
2. Provide HMI with the policies-related data for the named application that SDL sends within response to SDL.ActivateApp.

13.1.2 Request

13.1.2.1 Behavior

HMI must:

1. Send `SDL.ActivateApp` to SDL when the User requests to activate the application either form UI or by VR. Include the `appID` of the application being activated to the request is a must.

Note:

The information about the application (name, ID, etc.) is provided by SDL via either `BasicCommunication.UpdateAppList` or `BasicCommunication.OnAppRegistered` RPCs.

2. Wait for the response from SDL before activating the named application.
3. Read the information from SDL's response and behave in one of the following ways depending on parameters provided.

3.1. DO NOT activate the application if:

- 3.1.1. `isAppRevoked: true`
 - a) Do not activate the application
 - b) Request the appropriate user-friendly message from SDL (via `SDL.GetUserFriendlyMessage`
`{messageCodes: "AppUnauthorized"}`)
 - c) Unregister the corresponding application upon `BasicCommunication.OnAppUnregistered{appId}` (*Information: while the appId has NULL permissions in PT, SDL will reject such app's registration afterwards*)

Note:

Related diagram: see section #12.1.4.2.

3.2. Activate the application for the following cases:

- 3.2.1. `isSDLAllowed: true` and `isAppPermissionsRevoked: true`
 - a) Activate the application
 - b) Request the appropriate user-friendly message from SDL (via `SDL.GetUserFriendlyMessage`
`{messageCodes: "AppPermissionsRevoked" }}}`
 - c) Display the message to the User.

Note:

Related diagram: see section [#13.1.4.1](#)

- 3.2.2 `isSDLAllowed: true` and `isAppPermissionsRevoked: false`
 - a) Activate the application

13.1.2.1 App Launching and Querying (Open-source and Ford-specific)

Applications may support the feature of applications launching and querying. This option is available only to the applications which supports SDL 4.0 protocol.

This feature means that currently registered single SDL 4.0-enabled application may provide HMI with the list of applications available for running on HMI from the named device. HMI is informed via UpdateAppList request from SDL about the list of applications. The listed applications may be the not-yet-registered-with-SDL applications, still the User has the possibility to choose any of them and have it launched on the mobile device, registered with SDL, and activated then on vehicle HMI. In case the User-requested application fails to be launched on mobile device and/or to be registered, SDL will notify HMI with the appropriate resultCode (for more details see [13.1.4.3 diagram SDL.ActivateApp in AppLaunching and Querying sequence](#)).

HMI must:

- 1) Update applications list with appropriate applications data obtained via [UpdateAppList](#)
- 2) To grey out the application names (that is, make them impossible for activation) that come with "greyOut:true" parameter in [UpdateAppList](#).
Note: applications that come with "greyOut:false" parameter in [UpdateAppList](#) must be available for activation
- 3) To send SDL.ActivateApp to SDL in case the User chooses an app name from HMI (either by button press or by voice.)
- 4) Use progress bar during application registering to hide application registering process from the user and notify him about the app launching is in progress
- 5) Notify user with appropriate pop-up message when available-for-launching application selected in the list can't be launched
(APPLICATION_NOT_REGISTERED got from SDL as a response of SDL.ActivateApp)

[13.1.2.2 Parameters](#)

Param Name	Type	Mandatory	Additional	Description
appID	Integer	true	-	ID of the application that the User has chosen.

[13.1.3 Response](#)

Note:

There is a difference in message type for WebSocket and D-Bus connection (see the table below).

Result	Description	Message type		Message Params	Notes
		WebSocket	D-Bus		
Success	SUCCESS: SDL has processed the request and provides the app's policies status information.	JSON response	Method return	code: 0, params	'params' are described in section 12.1.3.1 Parameters.

Failure	INVALID_ID appID is invalid (e.g. the app with current id doesn't exist)	JSON error message		code:13	
	GENERIC_ERROR: The unknown issue occurred or other codes are not applicable.		Method return	code: 22	

13.1.3.1 Parameters

Param Name	Type	Mandatory	Additional	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	-	If isSDLAllowed is false, consent for sending PT through specified device is required.
isPermissionsConsentNeeded	Boolean	true	-	SDL must return always "false" only for Genivi: - 'false' , in case there are no policy groups that require User's consent.
isAppPermissionsRevoked	Boolean	true	-	SDL returns: - 'true' , in case the latest PT Update removed one or more permission groups from the named app's policies. - 'false' , in case there are no permission groups removed from the named app's policies.
appRevokedPermissions	Common.PermissionItem	false	array = true, minsize = 1, maxsize = 100	If app permissions were reduced (isAppPermissionsRevoked == true), then this array specifies list of removed permissions.
isAppRevoked	Boolean	true	-	SDL returns: - 'true' , in case the latest PT Update specified the named app to have NULL permissions (meaning the app became unauthorized). - 'false' , in case the named app has <u>different</u> from NULL permissions.
priority	Common.AppPriority	false	-	Send to HMI so that it can coordinate order of requests/notifications correspondingly.

13.1.3.2 AppPriority

Element name	Value	Short Description
--------------	-------	-------------------

Element name	Value	Short Description
EMERGENCY	0	
NAVIGATION	1	
VOICE_COMMUNICATION	2	
COMMUNICATION	3	
NORMAL	4	
NONE	5	

13.1.3.3 DeviceInfo Structure

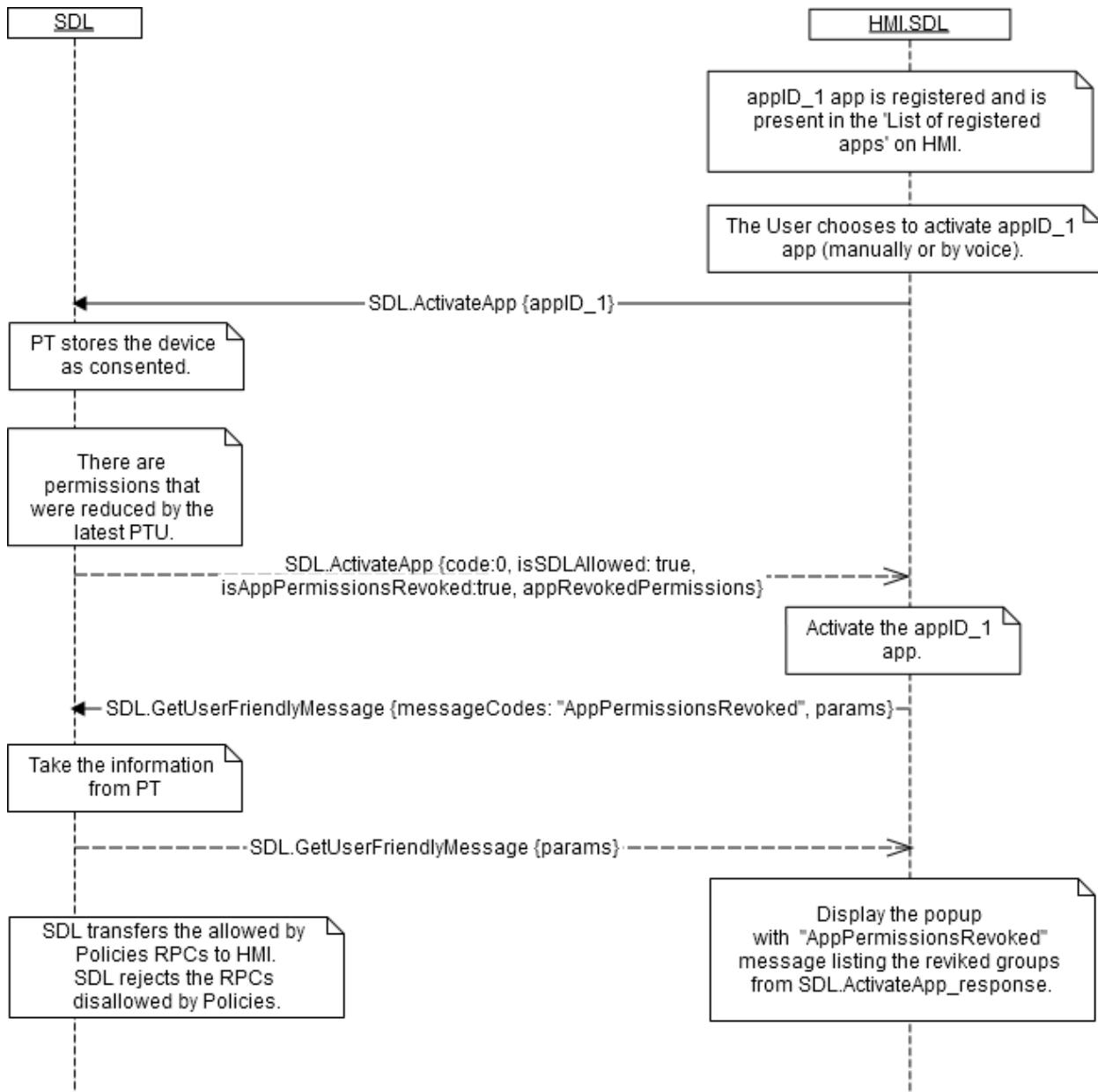
Param Name	Type	Mandatory	Additional	Description
name	String	true		The name of the device connected
id	String	true		The ID of the device connected. either hash of device's USB serial number (in case of USB connection) or hash of device's MAC address (in case of BlueTooth or WiFi connection).It remains unique between the ignition cycles for the same transport type .
transportType	Common.TransportType	false		The transport type the named-app's-device is connected over to HU (BlueTooth, USB or WiFi). Always returned by SDL in OnAppRegistered and UpdateAppList RPCs.
isSDLAllowed	Boolean	false		Sent by SDL in UpdateDeviceList. Always set-up to 'true' for Genivi (a device is allowed for PolicyTable Exchange)

13.1.3.4 TransportType Enumeration

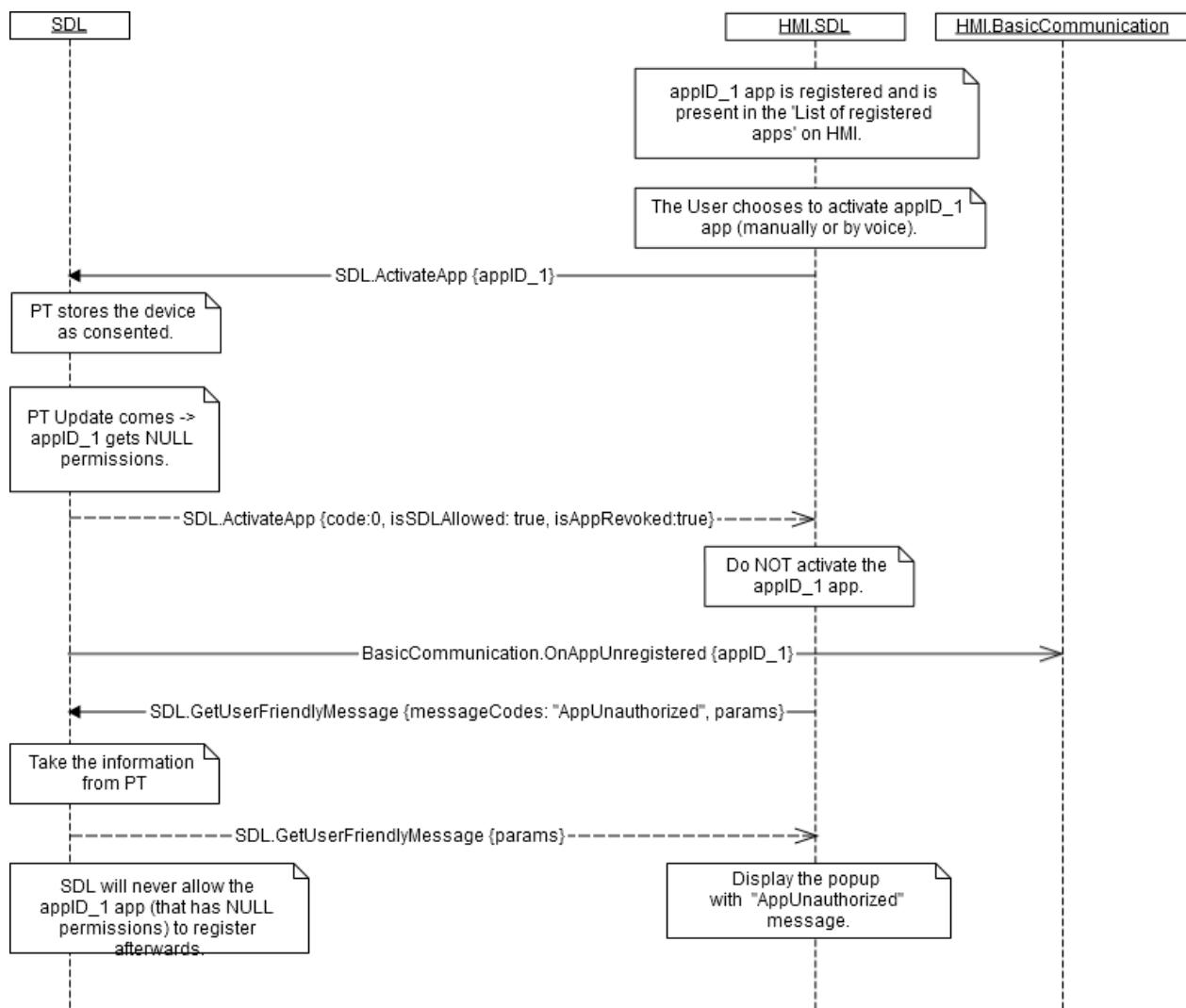
Element name	Value	Short Description
BLUETOOTH	0	
USB_IOS	1	
USB_AOA	2	
WIFI	3	

13.1.4 Sequence Diagrams

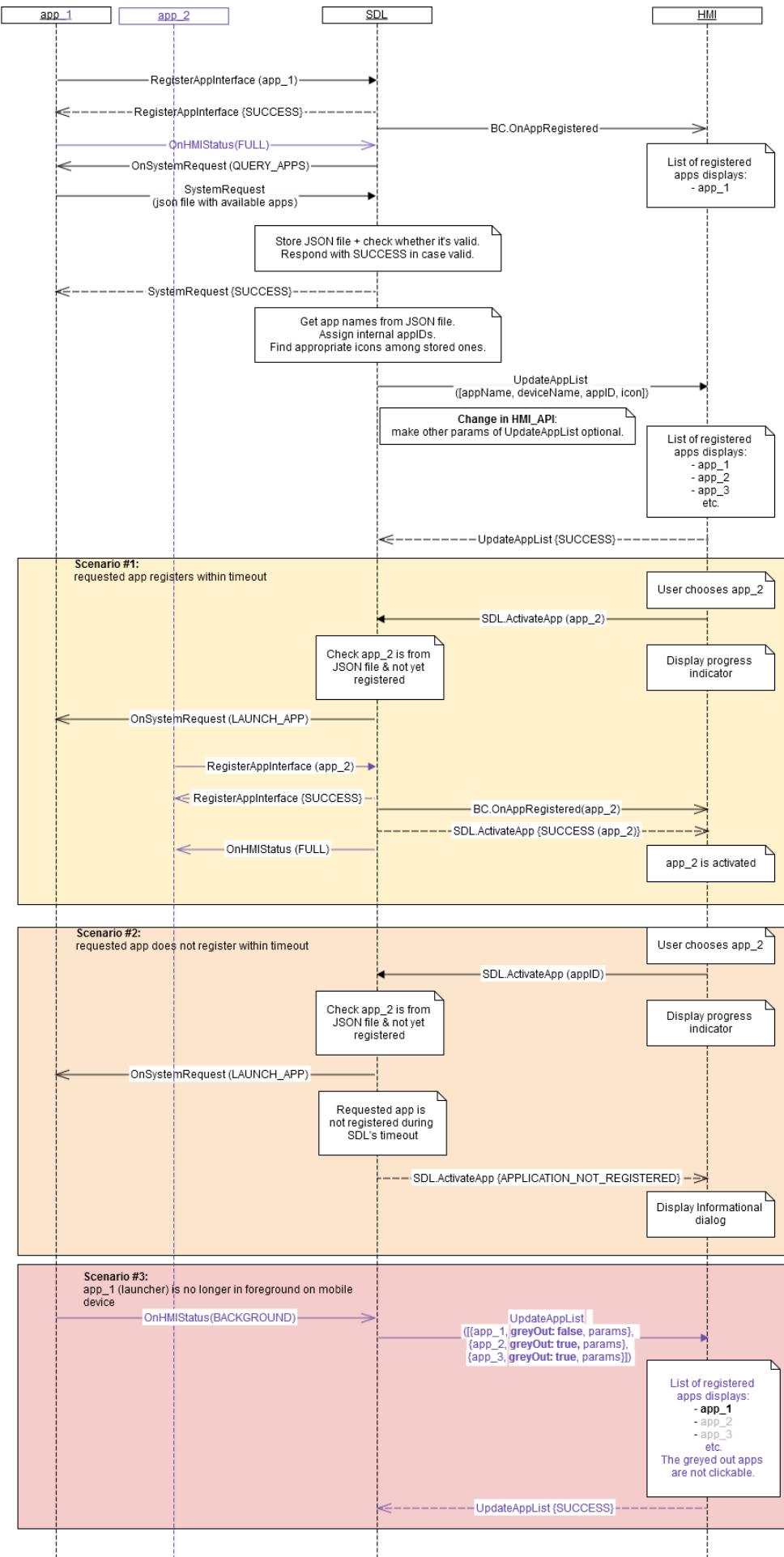
13.1.4.1 SDL.ActivateApp for the application registered and this application was reduced in permissions by the latest PT Update.



13.1.4.2 SDL.ActivateApp for the application registered and this application was unauthorized by the latest PT Update.



**13.1.4.3 *SDL.ActivateApp* in App Launching
and Querying sequence**



13.1.5 JSON Messages Examples

13.1.5.1 Request

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

13.1.5.2 Response

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

13.1.5.3 Error message

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

13.2 GetListOfPermissions

13.2.1 Description

Type:	Function
Sender:	HMI
Purpose:	Get the list of permissions either for the named application or for all applications.

The function returns the permission groups names the status of which (that is ‘allowed’ or ‘disallowed’) may be changed by the User.

13.2.2 Request

13.2.2.1 Behavior

HMI 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.
- 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.

Important: 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 [13.2.4.2 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 (fore more details see [13.8.2.2 OnAppPermissionConsent \(id<->name dependency\)](#))

Note:

The information about the application (name, ID, etc.) is provided by SDL via either BasicCommunication.UpdateAppList or BasicCommunication.OnAppRegistered RPCs.

13.1.2.1 Parameters

Param Name	Type	Mandatory	Additional	Description
appID	integer	false	-	ID of the application the list of permissions is required for. If the parameter is omitted, the permissions list will be returned for all currently registered applications on HMI

13.2.3 Response

Note:

There is a difference in message type for WebSocket and D-Bus connection (see the table below).

Result	Description	Message type		Message Params	Notes
		WebSocket	D-Bus		
Success	SUCCESS:	JSON response	Method return	allowedFunctions, code: 0	
Failure	INVALID_ID appID is invalid (e.g. the app with current id doesn't exist)			code: 13	
	INVALID_DATA The data sent is invalid (invalid JSON syntax, parameters out of bounds or of wrong type)	JSON error message	Method return	code: 11	Applicable for this RPC result codes.
	GENERIC_ERROR: The unknown issue occurred or other codes are not applicable.			code: 22	Please see Result Enumeration for all SDL-supported codes.

13.2.3.1 Parameters

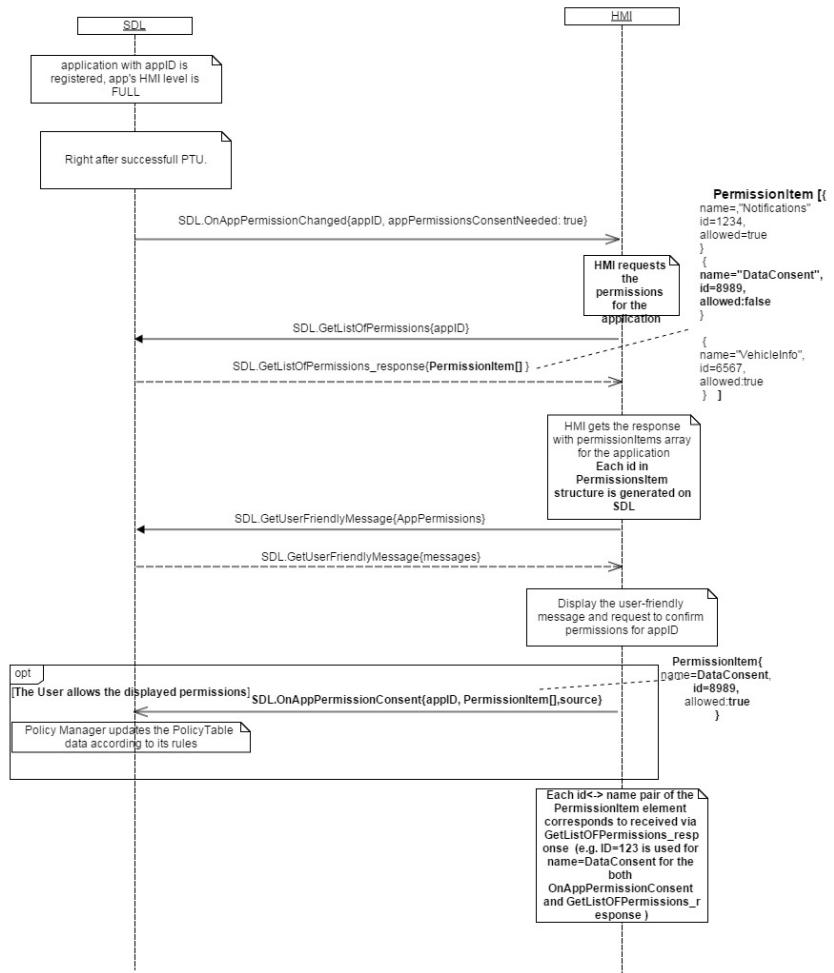
Param Name	Type	Mandatory	Additional	Description
allowedFunctions	Common.PermissionItem	true	array="true" minsize="0" maxsize="100"	If no permissions were specified for application the array will come empty.

13.2.3.2 PermissionItem

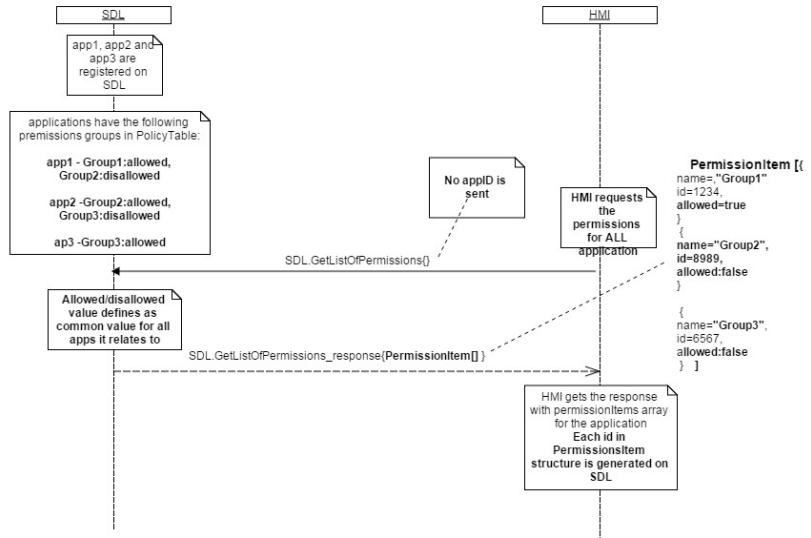
Param Name	Type	Mandatory	Description
name	String	true	
id	Integer	true	
allowed	Boolean	false	Specifies whether functionality was allowed/disallowed. If omitted - no information about User Consent is yet found for app.

13.2.4 Sequence Diagrams

13.2.4.1 GetListOfPermissions



13.2.4.2 GetListOfPermissions without appId



13.2.5 JSON Messages Examples

13.2.5.1 Request

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

13.2.5.2 Response

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

        [
            {
                "name" : "Location-1",
                "id":1234,
                "allowed":true
            },
            {
                "name" : "Notifications",
                "id":76876,
                "allowed":false
            }
        ]
    }
}
```

```

        },
    ]
}

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

13.2.5.3 Error message

```
{
    "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"
        }
    }
}
```

13.3 GetStatusUpdate

13.3.1 Description

Type:	Function
Sender:	HMI
Purpose:	Get information about current status of PT update process.

13.3.2 Request

13.3.2.1 Behavior

If HMI needs to get the current status of SDL, it may send GetStatusUpdate request to SDL.

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 allows to request the status of policy table at any time, not on update only.

HMI must:

- Send a request to HMI if it needs to get a current policy update status according to its workflows

13.3.3 Response

Note:

There is a difference in message type for WebSocket and D-Bus connection (see the table below).

Result	Description	Message type		Message Params	Notes
		WebSocket	D-Bus		
Success	SUCCESS:	JSON response	Method return	code: 0	
Failure	INVALID_DATA: Wrong JSON	JSON error message	Method return	code: 11	Applicable for this RPC result codes.
	GENERIC_ERROR: The unknown issue occurred or other codes are not applicable.			code: 22	Please see Result Enumeration for all SDL-supported codes.

13.3.3.1 Parameters

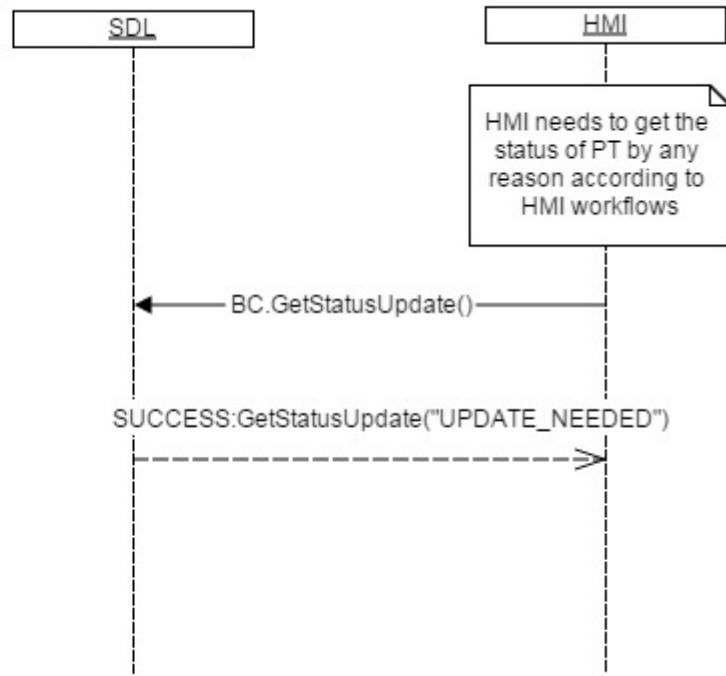
Param Name	Type	Mandatory	Additional	Description
status	Common.UpdateResult	true	-	See 13.3.3.2 UpdateResult

13.3.3.2 UpdateResult

Element	Description
UP_TO_DATE	PolicyTable is up to date and doesn't require an update
UPDATING	The process of Policy Table Update is in Progress
UPDATE_NEEDED	Policy Table requires to be updated

13.3.4 Sequence Diagrams

13.3.4.1 GetStatusUpdate



13.3.5 JSON Messages Examples

13.3.5.1 Request

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

13.3.5.2 Response

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

13.3.5.3 Error message

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

```

{
    "method" :
"SDL.GetStatusUpdate"
}
}
}

```

13.4 GetUserFriendlyMessage

13.4.1 Description

Type:	Function
Sender:	HMI
Purpose:	Get the user-friendly message(s) reserved in Policy Table

13.4.2 Request

13.4.2.1 Behavior

HMI 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

13.4.2.2 Parameters

Param Name	Type	Mandatory	Additional	Description
messageCodes	String	true	array="true" minsize="1" maxsize="100" maxlength="500"	Id of message to be received according to Policy Table i.e. StatusNeeded, Notifications, DrivingCharacteristics etc.
language	Common.Language	false	-	Optional parameter if HMI wants message in some other language then its current one already known to SDL.

13.4.2.3 MessageCodes

Message codes specify appropriate user messages which notifies 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.

Important: “MessageText” and “MessageCode” columns specify the messages just for an information purposes.

The MessageText received from SDL may differ from listed in the table.

MessageCode	MessageText
DrivingCharacteristics	An app can access the following driving characteristics: Fuel Consumption, MyKey, Seat Belt Status.
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
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.

13.4.3 Response

Note:

There is a difference in message type for WebSocket and D-Bus connection (see the table below).

Result	Description	Message type		Message Params	Notes
		WebSocket	D-Bus		
Success	SUCCESS:	JSON response	Method return	code: 0	
Failure	INVALID_DATA: Data is out of bounds or wrong JSON	JSON <u>error message</u>	Method return	code: 11	Applicable for this RPC result codes.
	GENERIC_ERROR: Some fail occurred or other codes are not applicable.			code: 22	Please see Result Enumeration for all SDL-supported codes.

13.4.3.1 Parameters

Param Name	Type	Mandatory	Additional	Description
messages	Common.UserFriendlyMessage	false	array="true" minsize="1" maxsize="100"	If no message was found in PT for specified message code and for HMI current or specified language, this parameter will be omitted.

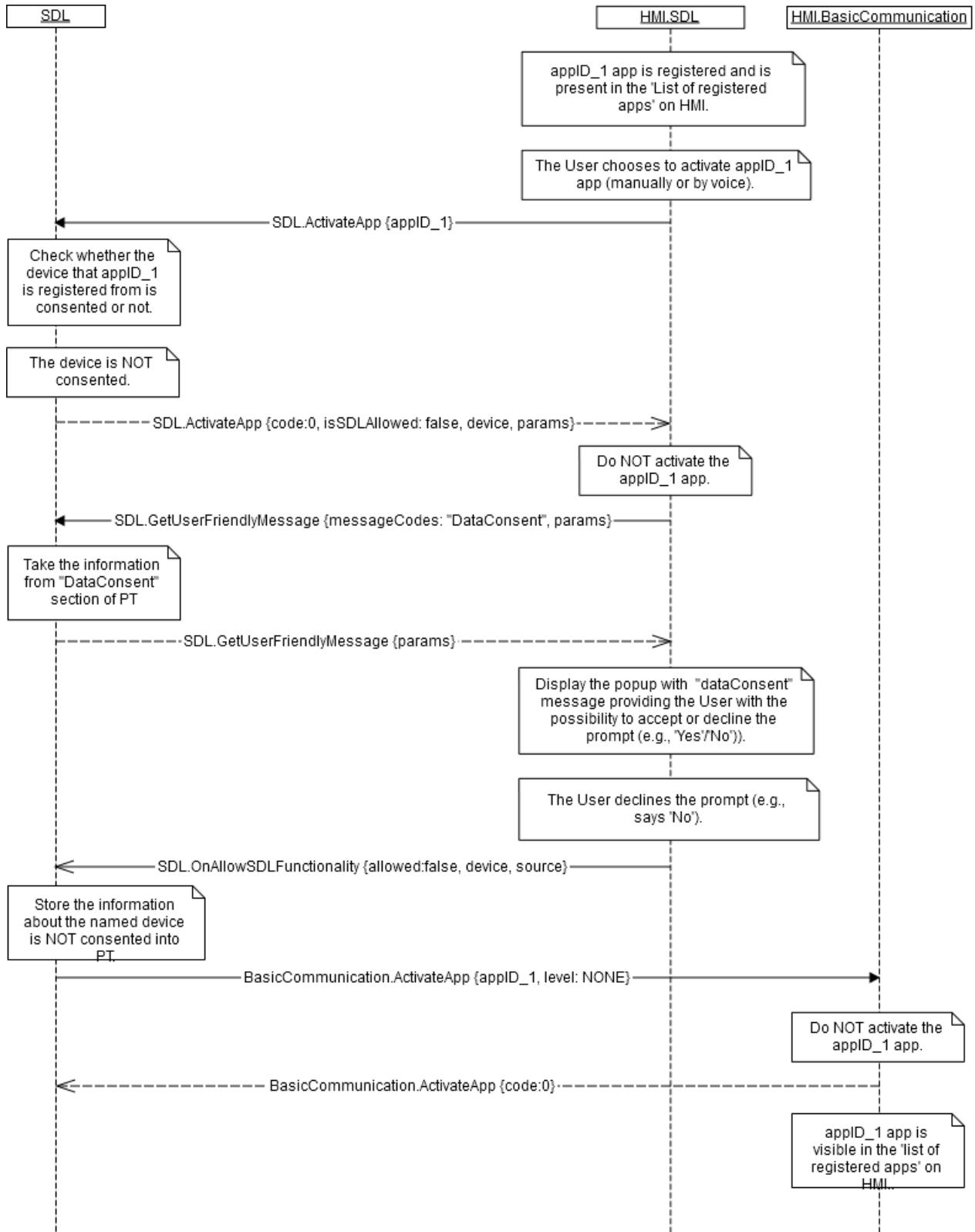
13.4.3.2 UserFriendlyMessage

Param Name	Type	Mandatory	Description
messageCode	String	true	Id of message to be received according to Policy Table i.e. StatusNeeded, Notifications, DrivingCharacteristics etc. See 13.4.2.3 MessageCodes for more details

Param Name	Type	Mandatory	Description
ttsString	String	false	
label	String	false	
line1	String	false	
line2	String	false	
textBody	String	false	

13.4.4 Sequence Diagrams

13.4.4.1 GetUserFriendlyMessage for DeviceConsent



13.4.5 JSON Messages Examples

13.4.5.1 Request

```
{
```

```

    "id" : 176,
    "jsonrpc" : "2.0",
    "method" :
"SDL.GetUserFriendlyMessage",
    "params" :
{
    "messageCodes": "AppPermissions",

    "language" : "EN-GB"
}

}

```

13.4.5.2 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"
}
}

```

13.4.5.3 Error message

```

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

```

13.5 OnAppPermissionChanged

13.9.1 Description

Type:	Notification
Sender:	SDL
Purpose:	Inform about the permissions changes for the application

Notification from SDL to HMI. Occurs when app permissions were changed. If no permission specified

means that app was disallowed and has to be unregistered.

Note: SDL sends the list of RequestTypes allowed by Policies via OnAppPermissionChanged API. In case HMI will use some not allowed by PolicyTable, SDL will ignore all notifications from HMI which contain prohibited RequestTypes values (e.g. sent via OnSystemRequest).

13.5.1.1 Parameters

Param Name	Type	Mandatory	Additional	Description
appID	Integer	true		
isAppPermissionsRevoked	Boolean	false		
appRevokedPermissions	Common.PermissionItem	false	array="true" minsize="1" maxsize="100"	If app permissions were reduced (isAppPermissionsRevoked == true), then this array specifies list of removed permissions
appRevoked	Boolean	false		If present then specified application was prohibited to used with Sync.
appPermissionsConsentNeeded	Boolean	false		If present specifies that permissions were added to application that require User Consent, then HMI can send GetListOfPermissions request to obtain list of permissions.
appUnauthorized	Boolean	false		When present and set to true (should be if present) then this means that application was not authorized (nickname check failed.)
priority	Common.AppPriority	false		Send to HMI so that it can coordinate order of requests/notifications correspondingly.
requestType	Common.RequestType	false	minsize="0" maxsize="100" array="true"	The list of SystemRequest's RequestTypes allowed by policies for the named application (the app's SystemRequest sent with RequestType out of this list will get 'disallowed' response from SDL). If SDL sends an empty array - any RequestType is allowed for this app. If SDL omits this parameter - nothing is changed for RequestType in the policies

13.5.1.3 AppPriority

Element name	Value	Short Description
--------------	-------	-------------------

Element name	Value	Short Description
EMERGENCY	0	
NAVIGATION	1	
VOICE_COMMUNICATION	2	
COMMUNICATION	3	
NORMAL	4	
NONE	5	

13.5.1.3 RequestType Enumeration

Element name	Value	Short 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	
MEDIA	18	
FOTA	19	

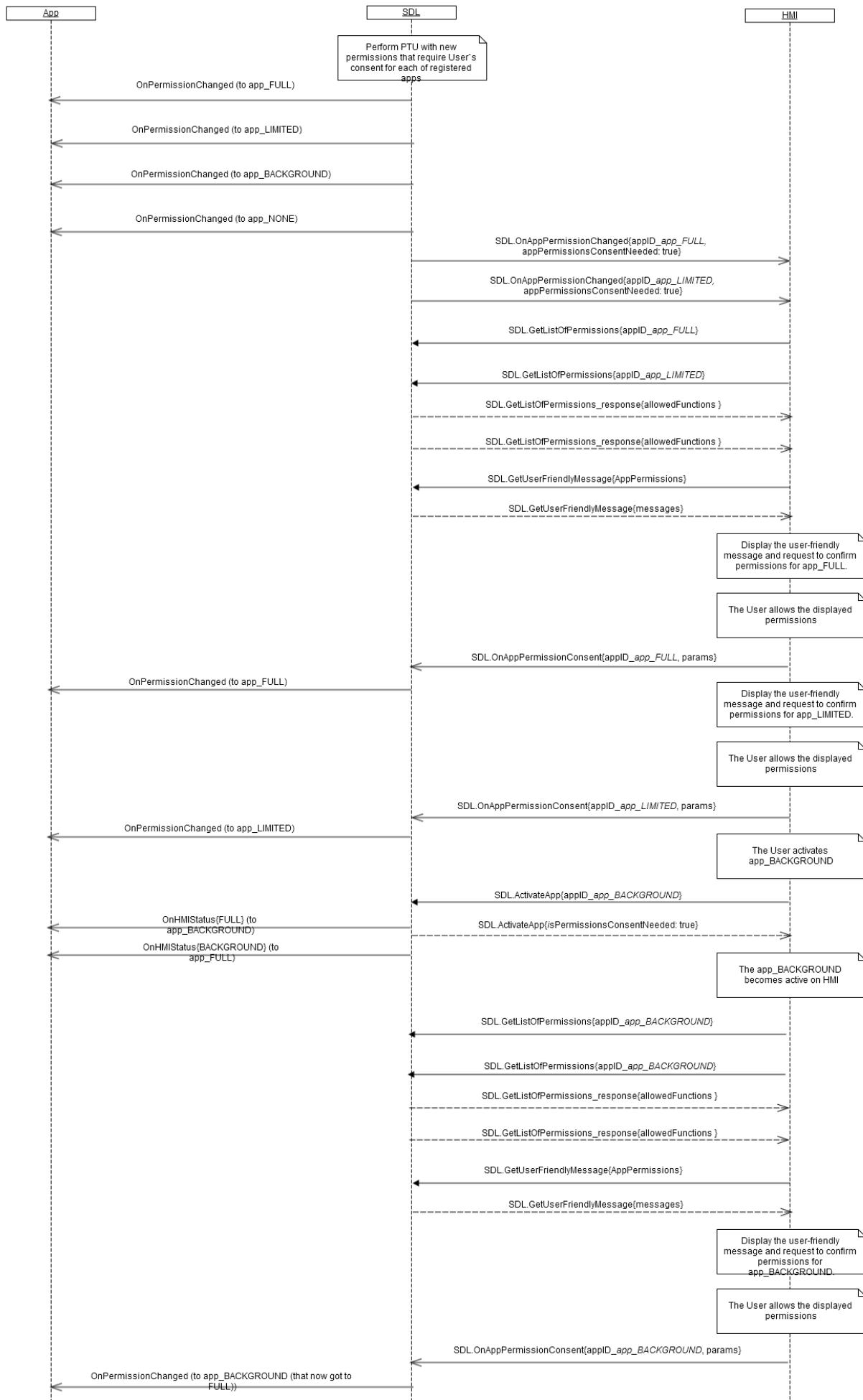
13.5.2 Sequence Diagrams

13.5.2.1 OnAppPermissionChanged with appPermissionsConsentNeeded:true @TODO to update

Pre-conditions 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.



13.5.3 JSON Messages Examples

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

13.6 OnStatusUpdate

13.6.1 Description

Type:	Notification
Sender:	SDL
Purpose:	Inform about the status of Policy Table update

PoliciesManager must notify HMI about Policy Table status update via SDL.OnStatusUpdate notification right after one of the statuses of UPDATING, UPDATE_NEEDED and UP_TO_DATE is changed from one to another.

13.6.1.1 Parameters

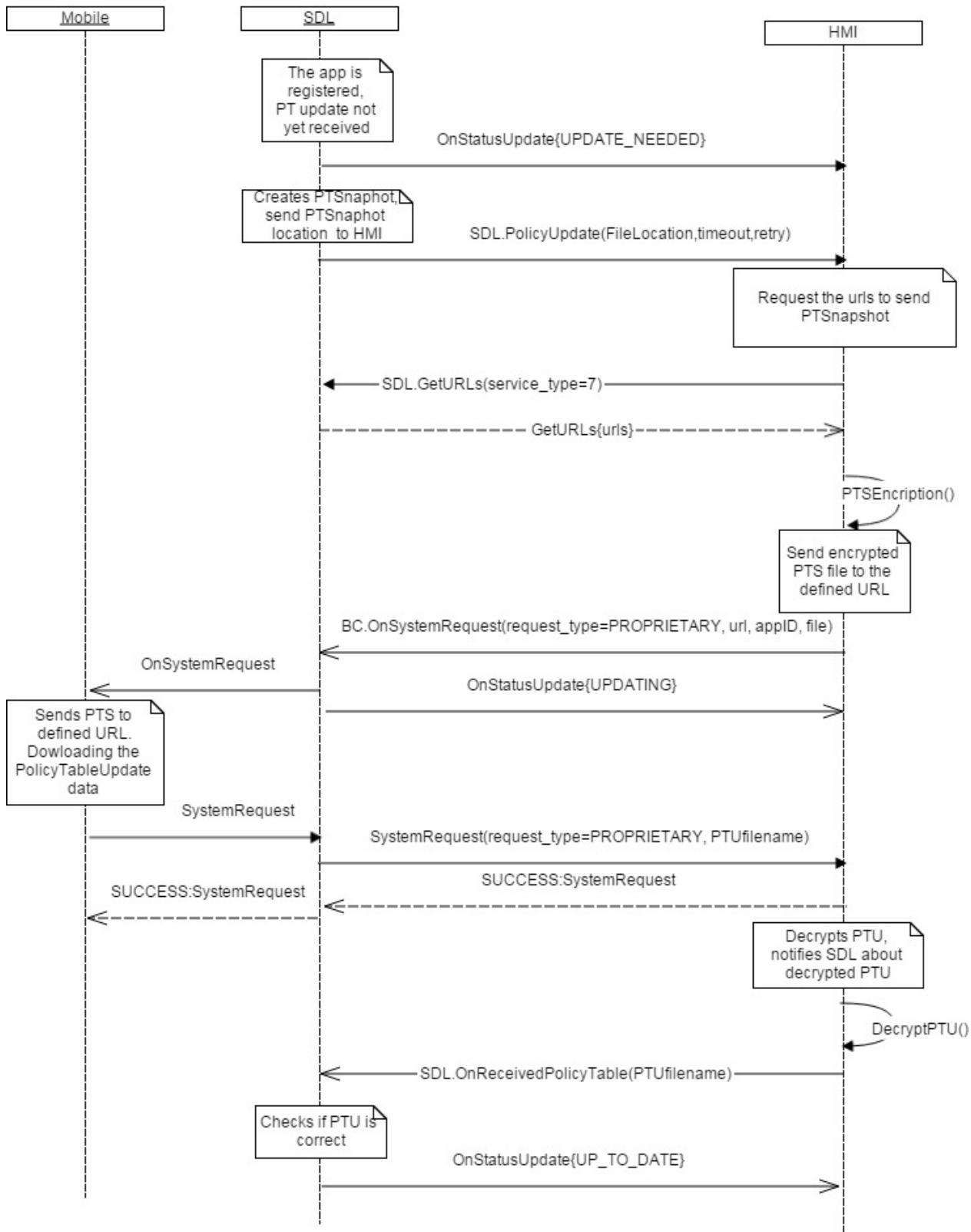
Param Name	Type	Mandatory	Description
status	Common.UpdateResult	true	Notification from SDL to HMI when current status of T exchange changed (i.e. requires an update, update in progress, up to date etc)

13.6.1.2 UpdateResult

Element	Description
UP_TO_DATE	Status is got right after applying the Updated PT delivered
UPDATING	Status got when PTSnapshot has just sent out to mobile app via OnSystemRequest RPC
UPDATE_NEEDED	PT requires an update. All the cases when PT requires an update are managed by SDL and Policy Manager (except of user manual update trigger)

13.6.2 Sequence Diagrams

13.6.2.1 OnStatusUpdate



13.6.3 JSON Messages Examples

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

14 Common Component Description

14.1 Enumerations

14.1.1 Result

The enumeration defines the possible result codes for all operations provided by API between SDL and HMI.

SUCCESS is the only result code that notifies that no errors, nonstandard situations or lack of resources/privileges were encountered.

Element Name	Value	Description
SUCCESS	0	The request is executed successfully.
UNSUPPORTED_REQUEST	1	The request is not supported.
UNSUPPORTED_RESOURCE	2	The requested resource is not supported under the current system. E.g.: notifications from the button are requested but physically this button is not present on HU.
DISALLOWED	3	API result code isn't applicable to HMI.

Element Name	Value	Description
REJECTED	4	<p>The requested command is rejected.</p> <p>E.g.:</p> <ul style="list-style-type: none"> - Because the mobile app is in background and cannot perform any commands from HMI. - Or an HU command (e.g. Speak) is rejected because a higher priority HU command (e.g. Alert) is playing.
ABORTED	5	<p>A requested command was aborted.</p> <p>E.g.:</p> <ul style="list-style-type: none"> - Due to user interaction (e.g. user pressed the button). - Or an HMI command (e.g. Speak) is aborted because a higher priority HMI command (e.g. Alert) was requested.
IGNORED	6	<p>A command was ignored, because the intended result is already in effect.</p> <p>E.g.:</p> <ul style="list-style-type: none"> - SetMediaClockTimer was used to pause the media clock although the clock is paused already.
RETRY	7	<p>The user interrupted the RPC and indicated to start over.</p> <p>E.g. : PerformAudioPassThru .</p>

Element Name	Value	Description
IN_USE	8	The data may not be changed, because it is currently in use. E.g.: when trying to delete a command set that is currently involved in an interaction.
DATA_NOT_AVAILABLE	9	The requested data is not available.
TIMED_OUT	10	Overlay reached the maximum timeout and closed.
INVALID_DATA	11	The data sent is invalid. E.g.: <ul style="list-style-type: none"> - Invalid Json syntax - Parameters out of bounds (number or enum range) - Mandatory parameters not provided - Parameter provided with wrong type - Invalid characters - Empty string
CHAR_LIMIT_EXCEEDED	12	The string data is too long.
INVALID_ID	13	One of the provided IDs is not valid.
DUPLICATE_NAME	14	There was a conflict with an already registered name.
WRONG_LANGUAGE	16	The requested language is currently not supported.
OUT_OF_MEMORY	17	The system could not process the request because the necessary memory RAM couldn't be allocated
TOO_MANY_PENDING_REQUESTS	18	There are too many requests pending (means, that the response cannot be delivered yet)
NO_APPS_REGISTERED	19	The request cannot be executed because no application interface has been registered.
WARNINGS	21	The RPC is executed successfully but one or more items have a warning or failure (e.g. Speak).

Element Name	Value	Description
GENERIC_ERROR	22	During the API call an unknown or unspecified within the current Result Enumeration error has occurred.
USER_DISALLOWED	23	RPC is included in a functional group explicitly blocked by the user.
TRUNCATED_DATA	24	The request (e.g. ReadDID) executed successfully but the data exceeded the platform maximum threshold and thus, only part of the data is available.

Related items:

[Table of Contents](#)

14.1.2 AppHMIType

The enumeration that lists possible types of the application.

Element name	Short Description
DEFAULT	The application of default type.
COMMUNICATION	The application for communication
MEDIA	The media application
MESSAGING	The application of messaging type
NAVIGATION	The application of navigation type
INFORMATION	The application of information type
SOCIAL	The application of social type
BACKGROUND_PROCESS	The application does not require displaying the information
TESTING	The application of testing type
SYSTEM	The application of system type

Related items:

HMIApplication – [section 14.2.1](#)

[Table of Contents](#)

14.1.3 DeactivateReason

This enumeration specifies the non-application functionality the User may switch to.

Element name	Short Description
AUDIO	The User has navigated to embedded audio player (radio, etc.)
PHONECALL	The User has navigated to make a call.
NAVIGATIONMAP	The User has navigated to embedded navigation screen.
PHONEMENU	The User has navigated to phone menu.
SYNCSETTINGS	The User has navigated to HU settings menu.
GENERAL	Other screens navigation apart from other mobile app.

Related items:

OnAppDeactivated – [section 6.12](#)

[Table of Contents](#)

14.1.4 ApplicationsCloseReason

This enumeration describes the possible reasons for exiting from all of registered applications.

Element name	Short Description
IGNITION_OFF	When the ignition is going to be off.
MASTER_RESET	When the master reset takes place.
FACTORY_DEFAULTS	When the factory default settings are chosen.
SUSPEND	On ACC key position

Related items:

OnExitAllApplications – [section 6.16](#)

[Table of Contents](#)

14.1.5 ClockUpdateMode

The enumeration describes how HMI must update the media clock timer. The details are in the table below.

Element name	Short Description
COUNTUP	HMI must start the media clock timer counting upwards, as in time elapsed, in increments of 1 second.
COUNTDOWN	HMI must start the media clock timer counting downwards, as in time remaining, in decrements of 1 second.
PAUSE	HMI must pause the media clock timer.
RESUME	HMI must resume the media clock timer. The timer must resume counting in whatever mode was in

Element name	Short Description
	effect before pausing (i.e. COUNTUP or COUNTDOWN).
CLEAR	HMI must clear the media clock timer (previously set through <code>setMediaClockTimer</code>).

Related Items:

`SetMediaClockTimer` – [section 7.13](#)

[Table of Contents](#)

14.1.6 SystemContext

Enumeration that describes possible contexts the application might be in on HU.

Element name	Short Description
MAIN	If there is currently no user interaction (user-initiated or app-initiated) with the head-unit, the application must be notified the the SystemContext is MAIN.
VRSESSION	Must be sent if there is a current user interaction that is VR-oriented (VR becomes active as a result of PTT or PerformInteraction).
MENU	Must be sent if HMI is currently displaying an in-application menu onscreen.
HMI_OBSCURED	Must be sent if HU is currently obscuring the application display either with a system or with another application overlay (except of Alert element).
ALERT	Must be sent if the Alert message is currently displayed onscreen.

Related items:

`OnSystemContext` – [section 7.22](#)

[Table of Contents](#)

14.1.7 DisplayType

HMI must send the information about the supported displays within [DisplayCapabilities](#) structure of the [UI.GetCapabilities](#) RPC. The display types are described in the enumeration below.

Element name	Short Description
CID	Center Information Display. This display type provides a 2-line x 20 character "dot matrix" display.

Element name	Short Description
TYPE2	TYPE II display. 1 line older radio head unit.
TYPE5	TYPE V display Old radio head unit.
NGN	Next Generation Navigation display.
GEN2_8_DMA	GEN-2, 8 inch display.
GEN2_6_DMA	GEN-2, 6 inch display.
MFD3	3 inch GEN1.1 display
MFD4	4 inch GEN1.1 display
MFD5	5 inch GEN1.1 display
GEN3_8-INCH	GEN-3, 8 inch display.

Related items:

DisplayCapabilities – [section 14.2.7](#)

GetCapabilities – [section 7.2](#)

[Table of Contents](#)

14.1.8 MediaClockFormat

The enumeration describes the format of media clock value supported by HMI.

Media clock is used by media applications for indicating the total/remaining duration of a song/album.

Upon UI.GetCapabilities request from SDL HMI must provide the information on supported media clock format sent within [DisplayCapabilities](#) structure. Further SDL will use the accurate values of the corresponding format for setting the time of media clock on UI.

Element name	Short Description
CLOCK1	<pre>minutesFieldWidth = 2; minutesFieldMax = 19; secondsFieldWidth = 2; secondsFieldMax = 99; maxHours = 19; maxMinutes = 59; maxSeconds = 59;</pre> <p>Is used for Type II, NGN and CID head units.</p>

Element name	Short Description
CLOCK2	<pre>minutesFieldWidth = 3; minutesFieldMax = 199; secondsFieldWidth = 2; secondsFieldMax = 99; maxHours = 59; maxMinutes = 59; maxSeconds = 59;</pre> <p>Is used for Type V head units.</p>
CLOCK3	<pre>minutesFieldWidth = 2; minutesFieldMax = 59; secondsFieldWidth = 2; secondsFieldMax = 59; maxHours = 9; maxMinutes = 59; maxSeconds = 59;</pre> <p>Is used for GEN1.1 (i.e. MFD3/4/5) head units.</p>
CLOCKTEXT1	<p>5 characters possible</p> <p>Format: 1 sp c : sp c c 1 sp : digit "1" or space c : character out of following character set: sp 0-9 [letters : sp : colon or space</p> <p>Is used for Type II head unit</p>
CLOCKTEXT2	<p>5 chars possible</p> <p>Format: 1 sp c : sp c c 1 sp : digit "1" or space c : character out of following character set: sp 0-9 [letters : sp : colon or space</p> <p>Is used for CID and NGN head unit.</p>
CLOCKTEXT3	<p>6 chars possible</p> <p>Format: 1 sp c c : sp c c 1 sp : digit "1" or space c : character out of following character set: sp 0-9 [letters : sp : colon or space</p> <p>Is used for Type V head unit.</p>
CLOCKTEXT4	<p>6 chars possible</p> <p>Format: c : sp c c : c c : sp : colon or space c : character out of following character set: sp 0-9 [letters].</p> <p>Is used for GEN1.1 (i.e. MFD3/4/5) head units.</p>

Related items:

DisplayCapabilities – [section 14.2.7](#)

TextFieldName – [section 14.1.15](#)

[Table of Contents](#)

14.1.9 HmiZoneCapabilities

Upon [GetCapabilities](#) request, HMI must provide the information about the zone it is available from:

Element name	Short Description
FRONT	Indicates HMI available for front seat passengers.
BACK	Indicates HMI available for rear seat passengers.

Related items:

GetCapabilities – for UI, [section 7.2](#)

[Table of Contents](#)

14.1.10 SpeechCapabilities

Upon TTS .GetCapabilities request, HMI must provide the information about the zone it is available from:

Element name	Short Description
TEXT	
SAPI_PHONEMES	
LHPLUS_PHONEMES	
PRE_RECORD	
SILENCE	

Related items:

GetCapabilities – for TTS, [section 10.2](#)

[Table of Contents](#)

14.1.11 VrCapabilities

Upon TTS .GetCapabilities request, HMI must provide the information about the zone it is available from:

Element name	Short Description

Element name	Short Description
TEXT	

Related items:

GetCapabilities – for VR, [section 9.2](#)

[Table of Contents](#)

14.1.12 DriverDistractionState

HMI must provide the information about either the driver distraction rules are in effect or not. This enumeration describes the possible states of driver distraction.

Element name	Short Description
DD_ON	Driver Distraction rules are in effect.
DD_OFF	Driver Distraction rules are not in effect.

Related items:

OnDriverDistraction – [section 7.27](#)

[Table of Contents](#)

14.1.11 SoftButtonType

The enumeration defines the types of the soft buttons to be displayed on UI component:

- 1) The text is displayed on the soft button
- 2) The image is displayed on the soft button
- 3) Both image and text are displayed on the soft button.

Element name	Short Description
TEXT	Text displayed
IMAGE	Image displayed
BOTH	Both text and image displayed

Related items:

SoftButton – [section 14.2.13](#)

[Table of Contents](#)

14.1.12 SystemAction

The HMI may provide three possible ways of reacting on soft button click (the exact behavior may vary and be platform-dependent):

- 1) Default action (DEFAULT_ACTION). For example, if the soft button is preset for cleaning the overlay, the click on it should close the pop-up or clean the overlay.
- 2) Returning application to the full mode (STEAL_FOCUS). The soft button with presets for this action should close the pop-up window and return the application which caused that pop-up to the full mode.
- 3) Making the pop-up or event remain active (KEEP_CONTEXT). The pop-up or event should be left on the screen after clicking the soft button with presets for this action, renewing the timeout for this event.

The responsibility of behavior on System Action implementation is on HMI.

If HMI does not support some System Action it has to respond correspondingly.

The following enumeration defines the system actions for the soft button click:

Element name	Short Description
DEFAULT_ACTION	Default action should occur. Standard behavior (e.g. SoftButton clears overlay)
STEAL_FOCUS	- The pop-up/dialog should be closed - The app, having been obscured with that pop-up/dialog, should be returned into the full mode.
KEEP_CONTEXT	- The pop-up/dialog should be left on the screen - The timeout for this pop-up/dialog should be renewed

Related items:

SoftButton – [section 14.2.13](#)

[Table of Contents](#)

14.1.13 TextAlignement

This enumeration lists the variants of text alignment.

Element name	Short Description
LEFT_ALIGNED	Text is aligned left.
RIGHT_ALIGNED	Text is aligned right.

Element name	Short Description
CENTERED	Text is centered.

Related items:

Show – [section 7.5](#)

[Table of Contents](#)

14.1.14 Language

The enumeration defines the possible languages that the HU components might support and the SDL might request:

Element name	Short Description
AR-SA	Arabic – Saudi Arabia
CS-CZ	Czech – Czech Republic
DA-DK	Danish – Denmark
DE-DE	German – Germany
EN-AU	English – Australia
EN-GB	English – GB
EN-US	English – US
ES-ES	Spanish – Spain
ES-MX	Spanish – Mexico
FR-CA	French – Canada
FR-FR	French – France
IT-IT	Italian – Italy
JA-JP	Japanese – Japan
KO-KR	Korean – South Korea
NL-NL	Dutch (Standard) – Netherlands
NO-NO	Norwegian - Norway

Element name	Short Description
PL-PL	Polish – Poland
PT-PT	Portuguese – Portugal
PT-BR	Portuguese – Brazil
RU-RU	Russian - Russia
SV-SE	Swedish – Sweden
TR-TR	Turkish – Turkey
ZH-CN	Mandarin – China
ZH-TW	Mandarin – Taiwan
NL-BE	Dutch Belgium (Flemish)
EL-GR	Greek
HU-HU	Hungarian
FI-FI	Finnish
SK-SK	Slovak

Related items:

HMIApplication – [section 14.2.1](#)

GetSupportedLanguages – for UI, [section 7.3](#)

- for VR, [section 9.3](#)
- for TTS, [section 10.3](#)

ChangeRegistration – for UI, [section 7.13](#)

- for VR, [section 9.6](#)
- for TTS, [section 10.6](#)

GetLanguage – for UI, [section 7.14](#)

- for VR, [section 9.7](#)
- for TTS, [section 10.7](#)

OnLanguageChange – for UI, [section 7.23](#)

- for VR, [section 9.11](#)
- for TTS, [section 10.10](#)

[Table of Contents](#)

14.1.15 TextFieldName

The enumeration describes the possible fields for the text information to be displayed.

Upon SDL's request HMI must provide the list of supported fields.

Further SDL will request displaying the text information in these fields within different requests to HMI.

The table below contains the information about:

- The name of the field
- Conditions of displaying the text
- The name of the method concerned

Element name	Short Description
mainField1	<p>The text that must be displayed in a single or upper display line. If this value is not set, the text of mainField1 must stay unchanged. If this text is empty "", the field must be cleared. Applies to Show, section 7.5</p>
mainField2	<p>The text that must be displayed on the second display line. If this text is not set, the text of mainField2 must stay unchanged. If this text is empty "", the field must be cleared. Applies to Show, section 7.5</p>
mainField3	<p>The text that must be displayed on the second "page" first display line. If this text is not set, the text of mainField3 must stay unchanged. If this text is empty "", the field must be cleared. Applies to Show, section 7.5</p>
mainField4	<p>The text that must be displayed on the second "page" second display line. If this text is not set, the text of mainField4 must stay unchanged. If this text is empty "", the field must be cleared.</p>

Element name	Short Description
	Applies to Show, section 7.5
statusBar	<p>The text is placed in the status bar area.</p> <p>Note: This relates to navigation displays</p> <p>If this parameter is omitted, the status bar text must remain unchanged.</p> <p>If this parameter is an empty string, the field must be cleared.</p> <p>If provided and the display has no status bar, this parameter must be ignored.</p> <p>Applies to Show, section 7.5</p>
mediaClock	<p>Text value for MediaClock field. Shall arrive in the form as described in the MediaClockFormat enumeration</p> <p>If this text is set, any automatic media clock updates previously set with SetMediaClockTimer must be stopped.</p> <p>Applies to Show, section 7.5</p>
mediaTrack	<p>The text that should be displayed in the track field. This field should be valid only for media applications on.</p> <p>If this text is not set, the text of mediaTrack must stay unchanged.</p> <p>If this text is empty "", the field must be cleared.</p> <p>Applies to Show, section 7.5</p>
alertText1	<p>The text that must be displayed in the top field of the display during the Alert.</p> <p>Applies to Alert, section 7.4.</p>
alertText2	<p>The text that must be displayed in the bottom field of the display during</p>

Element name	Short Description
	the Alert. Applies to Alert, section 7.4.
alertText3	The optional third line of the alert text field. Applies to Alert, section 7.4.
scrollableMessageBody	The long form body of text that can include newlines and tabs. Applies to ScrollableMessage, section 7.17.
initialInteractionText	Must be displayed when the interaction begins. The text must be displayed on the first line of a multiline display, and must be centered. Applies to PerformInteraction, section 7.10.
navigationText1	The text that must be displayed on the first line of navigation text. Applies to ShowConstantTBT, section 12.3.
navigationText2	The text that must be displayed on the second line of navigation text. Applies to ShowConstantTBT, section 12.3.
ETA	Estimated Time of Arrival for navigation. Applies to ShowConstantTBT, section 12.3.
totalDistance	Total distance to destination for navigation. Applies to ShowConstantTBT, section 12.3.
navigationText	Navigation text for UpdateTurnList. Applies to Turn, section 14.2.33.
audioPassThruDisplayTe xt1	The first line of text that must be displayed during audio capture. Applies to PerformAudioPassTh ru, section 7.18.

Element name	Short Description
audioPassThruDisplayText2	<p>The second line of text that must be displayed during audio capture.</p> <p>Applies to PerformAudioPassThru, section 7.18.</p>
sliderHeader	<p>The text that must be displayed on the header of slider.</p> <p>Applies to Slider, section 7.16.</p>
sliderFooter	<p>The text that must be displayed on the footer of slider.</p> <p>Applies to Slider, section 7.16.</p>
notificationText	<p>The text that must be displayed to notify the User on some event.</p> <p>Applies to ShowNotification, not used now</p>
locationName	<p>Optional name / title of intended location for SendLocation, section 12.10</p>
locationDescription	<p>Optional description of intended location / establishment (if applicable) for SendLocation, section 12.10</p>
addressLines	<p>Optional location address (if applicable) for SendLocation, section 12.10</p>
phoneNumber	<p>Optional phone number of intended location / establishment (if applicable) for SendLocation, section 12.10</p>
menuName	Primary text for Choice
secondaryText	Secondary text for Choice
tertiaryText	Tertiary text for Choice
timeToDestination	<p>Total time to arrive a destination for navigation.</p> <p>Applies to ShowConstantTBT, section 12.3</p>

Element name	Short Description
turnText	To be removed

Related Items:

MediaClockFormat – [section 14.1.8](#)
 DisplayCapabilities – [section 14.2.7](#)
 TextFieldStruct – [section 14.2.11](#)
 Turn – [section 14.2.33](#)
 Alert – [section 7.4](#)
 Show – [section 7.5](#)
 PerformInteraction – [section 7.10](#)
 Slider – [section 7.16](#)
 ScrollableMessage – [section 7.17](#)
 PerformAudioPassThru – [section 7.18](#)
 ShowConstantTBT – [section 12.3](#)
 SendLocation – [section 12.10](#)

[Table of Contents](#)

14.1.17 ImageFieldName

This enumeration contains the information about the type of the image.

Element name	Short Description
softButtonImage	The image field for SoftButton
choiceImage	The first image field for Choice
choiceSecondaryImage	The secondary image field for Choice
vrHelpItem	The image field for vrHelpItem
turnIcon	The image field for Turn
menuIcon	The image field for the menu icon in SetGlobalProperties
cmdIcon	The image field for AddCommand
graphic	The image field for Show
showConstantTBTIcon	The primary image field for ShowConstantTBT
showConstantTBTNextTurnIcon	The secondary image field for ShowConstantTBT

Element name	Short Description
nextTurnIcon	

Related items:

DisplayCapabilities – section 14.2.7

Image – [section 14.2.15](#)

[Table of Contents](#)

14.1.18 ImageType

This enumeration contains the information about the type of the image.

Element name	Short Description
STATIC	Static image. The image that is sent as the binary or hex code within the request.
DYNAMIC	Dynamic image. The image that is stored on HMI and just a link to it is further used within requests.

Related items:

DisplayCapabilities – section 14.2.7

Image – [section 14.2.15](#)

[Table of Contents](#)

14.1.17 ButtonName

The following enumeration defines the hard (physical) and soft (touchscreen) buttons that may be available from HMI.

Element name	Short Description
OK	Represents the button usually labeled "OK". A typical use of this button is for the User to press it to make a selection.
SEEKLEFT	Represents the seek-left button. A typical use of this button is for the user to scroll to the left through menu choices, one menu item per press.
SEEKRIGHT	Represents the seek-right button. A typical use of this button is for the user to scroll to the right through menu choices one menu item per press.
TUNEUP	Represents a turn of the tuner knob in the clockwise direction one tick.

Element name	Short Description
TUNEDOWN	Represents a turn of the tuner knob in the counter-clockwise direction one tick.
PRESET_0	Represents the preset 0 button.
PRESET_1	Represents the preset 1 button.
PRESET_2	Represents the preset 2 button.
PRESET_3	Represents the preset 3 button.
PRESET_4	Represents the preset 4 button.
PRESET_5	Represents the preset 5 button.
PRESET_6	Represents the preset 6 button.
PRESET_7	Represents the preset 7 button.
PRESET_8	Represents the preset 8 button.
PRESET_9	Represents the preset 9 button.
CUSTOM_BUTTON	Represents any of touchscreen buttons provided by Mobile Application
SEARCH	Represents the 'Search' button.
PLAY_PAUSE	Represents Play/Pause button functionality (e.g. Play and Pause audio source)

Related items:

ButtonCapabilities – [section 14.2.15](#)
 PresetBankCapabilities – [section 14.2.16](#)
 GetCapabilities – for Buttons, [section 8.1](#)
 OnButtonEvent – [section 8.2](#)
 OnButtonPress – [section 8.3](#)

[Table of Contents](#)

14.1.18 ButtonEventMode

The following events must be supported by HMI for the hardware/soft buttons and must be reported within [UI.GetCapabilities](#) or [Buttons.GetCapabilities](#) RPCs:

- 2) The User has pressed the button (BUTTONDOWN event).
- 3) The User has released the button (BUTTONUP event).

HMI must provide the [OnButtonEvent](#) notification whenever the button is pressed/released.

This enumeration defines UP/DOWN events for the button.

Element name	Short Description
BUTTONDOWN	The button has been pressed
BUTTONUP	The button has been released

Related Items:

SoftButtonCapabilities – [section 14.2.12](#)

ButtonCapabilities – [section 14.2.15](#)

OnButtonEvent – [section 8.2](#)

[Table of Contents](#)

14.1.19 ButtonPressMode

The following press modes might be supported by HMI's hardware/soft buttons and if supported must be reported within [UI.GetCapabilities](#) or [Buttons.GetCapabilities](#) RPCs.

- 1) **Short** – occurs when a button is pressed, then released within two (varies depending on HMI) seconds. The event is considered to occur immediately after the button is released.
- 2) **Long** – occurs when a button is pressed and held for two seconds or more. The event is considered to occur immediately after the two (varies depending on HU) seconds threshold has been crossed, before the button is released

HMI must provide the [onButtonPress](#) notification whenever the Long/Short press occurs.

The following enumeration defines SHORT/LONG modes for the hardware/soft button:

Element name	Short Description
SHORT	Short-time button press
LONG	Long-time button press

Related Items:

SoftButtonCapabilities – [section 14.2.12](#)

ButtonCapabilities – [section 14.2.15](#)

OnButtonPress – [section 8.3](#)

[Table of Contents](#)

14.1.20 LayoutMode

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

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

Related Items:

UI.PerformInteraction - [section 7.10](#)

[Table of Contents](#)

14.1.21 TouchEvent

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

Element name	Short Description
TOUCHSTART	
TOUCHMOVE	
TOUCHEND	
DOUBLETOUCH	

Related Items:

OnTouchEvent - [section 7.25](#)

[Table of Contents](#)

14.1.22 SamplingRate

Describes different sampling options for PerformAudioPassThru

Element name	Short Description
8KHZ	
16KHZ	
22KHZ	
44KHZ	

Related Items:

PerformAudioPassThru, [section 7.18](#)

[Table of Contents](#)

14.1.23 BitsPerSample

Describes different quality options for PerformAudioPassThru.

Element name	Short Description
8_BIT	
16_BIT	

Related Items:

PerformAudioPassThru - [section 7.18](#)

[Table of Contents](#)

14.1.24 AudioType

Describes different audio type options for PerformAudioPassThru

Element name	Short Description
PCM	

Related Items:

PerformAudioPassThru, [section 7.18](#)

[Table of Contents](#)

14.1.25 KeyboardLayout

Enumeration listing possible keyboard layouts

Element name	Short Description
QWERTY	
QWERTZ	
AZERTY	

Related Items:

[UI.SetGlobalProperties – section 7.12](#)

[Table of Contents](#)

14.1.26 KeyboardEvent

Enumeration listing possible keyboard events.

Element name	Short Description
KEYPRESS	
ENTRY_SUBMITTED	
ENTRY_VOICE	Indicates that a voice button was pressed for a request to search via voice.
ENTRY_CANCELLED	
ENTRY_ABORTED	

Note: ENTRY_VOICE is **not** related to UI.PerformInteraction(KEYBOARD).

Related Items:

[OnKeyboardInput – section 7.24](#)

[Table of Contents](#)

14.1.27 KeypressMode

Enumeration listing possible keyboard events.

Element name	Short Description
SINGLE_KEYPRESS	Each keypress is individually sent as the user presses the keyboard keys.
QUEUE_KEYPRESSES	The keypresses are queued and a string is eventually sent once the user chooses to submit their entry.

RESEND_CURRENT_ENTRY	The keypresses are queued and a string is sent each time the user presses a keyboard key; the string contains the entire current entry.
----------------------	---

Related Items:

[UI.SetGlobalProperties – section 7.12](#)

[Table of Contents](#)

14.1.28 AmbientLightStatus

Reflects the status of the ambient light sensor.

Element name	Short Description
NIGHT	
TWILIGHT_1	
TWILIGHT_2	
TWILIGHT_3	
TWILIGHT_4	
DAY	
UNKNOWN	
INVALID	

Related Items:

[GetVehicleData – section 11.8](#)

[OnVehicleData – section 11.9](#)

[Table of Contents](#)

14.1.29 ECallConfirmationStatus

Reflects the status of the eCall Notification.

Element name	Short Description
NORMAL	
CALL_IN_PROGRESS	
CALL_CANCELLED	

CALL_COMPLETED	
CALL_UNSUCCESSFUL	
ECALL_CONFIGURED_OFF	
CALL_COMPLETE_DTMF_TIMEOUT	

Related Items:

[GetVehicleData - section 11.8](#)

[OnVehicleData - section 11.9](#)

[Table of Contents](#)

14.1.30 VehicleDataNotificationStatus

Reflects the status of a vehicle data notification.

Element name	Short Description
NOT_SUPPORTED	
NORMAL	
ACTIVE	
NOT_USED	

Related Items:

[GetVehicleData - section 11.8](#)

[OnVehicleData - section 11.9](#)

[Table of Contents](#)

14.1.31 EmergencyEventType

Reflects the emergency event status of the vehicle.

Element name	Short Description
NO_EVENT	
FRONTAL	
SIDE	
REAR	

ROLLOVER	
NOT_SUPPORTED	
FAULT	

Related Items:

[GetVehicleData - section 11.8](#)

[OnVehicleData - section 11.9](#)

[Table of Contents](#)

14.1.32 FuelCutoffStatus

Reflects the status of the RCM fuel cutoff.

Element name	Short Description
TERMINATE_FUEL	
NORMAL_OPERATION	
FAULT	

Related Items:

[GetVehicleData - section 11.8](#)

[OnVehicleData - section 11.9](#)

[Table of Contents](#)

14.1.33 PowerModeQualificationStatus

Reflects the status of the current power mode qualification.

Element name	Short Description
POWER_MODE_UNDEFINED	
POWER_MODE_EVALUATION_IN_PROGRESS	
NOT_DEFINED	
POWER_MODE_OK	

Related Items:

[GetVehicleData - section 11.8](#)

[OnVehicleData – section 11.9](#)

[Table of Contents](#)

14.1.34 CarModeStatus

Reflects the status of the current car mode.

Element name	Short Description
NORMAL	
FACTORY	
TRANSPORT	
CRASH	

Related Items:

[GetVehicleData – section 11.8](#)

[OnVehicleData – section 11.9](#)

[Table of Contents](#)

14.1.35 PowerModeStatus

Reflects the status of the current power mode.

Element name	Short Description
KEY_OUT	
KEY_RECENTLY_OUT	
KEY_APPROVED_0	
POST_ACCESSION_0	
ACCESSION_1	
POST IGNITION_1	
IGNITION_ON_2	
RUNNING_2	
CRANK_3	

Related Items:

[GetVehicleData – section 11.8](#)

[OnVehicleData – section 11.9](#)

[Table of Contents](#)

14.1.36 ComponentVolumeStatus

The enumeration provides the cases for the component volume status (e.g., fuel level, tire pressure).

Element name	Short Description
UNKNOWN	The data is unknown.
NORMAL	The volume is normal.
LOW	The volume is low.
FAULT	The module/sensor is currently faulted.
ALERT	The component's volume is in critical level.
NOT_SUPPORTED	The data is not supported.

Related items:

SingleTireStatus – [section 14.2.27](#)

[GetVehicleData – section 11.8](#)

[OnVehicleData – section 11.9](#)

[Table of Contents](#)

14.1.37 PRNDL

The following enumeration describes the possible positions of vehicle's change-speed lever.

Element name	Short Description
PARK	Parking
REVERSE	Reverse gear
NEUTRAL	No gear
DRIVE	Drive Sport mode
SPORT	Sport mode
LOWGEAR	1st gear hold
FIRST	The change-speed lever is in the first position
SECOND	The change-speed lever is in the second position
THIRD	The change-speed lever is in the third position
FOURTH	The change-speed lever is in the fifth position
FIFTH	The change-speed lever is in the sixth position
SIXTH	The change-speed lever is in the seventh position

Element name	Short Description
SEVENTH	
EIGHTH	
FAULT	

Related items:

[GetVehicleData – section 11.8](#)

[OnVehicleData – section 11.9](#)

[Table of Contents](#)

14.1.38 WarningLightStatus

The enumeration reflects the status of a cluster instrument warning light.

Element name	Short Description
OFF	The warning light is off
ON	The warning light is on
FLASH	The warning light is flashing
NOT_USED	

Related items:

TireStatus – [section 14.2.26](#)

[GetVehicleData – section 11.8](#)

[OnVehicleData – section 11.9](#)

[Table of Contents](#)

14.1.39 VehicleDataEventStatus

The following enumeration reflects the status of a vehicle data event (e.g. a seat belt event status).

Element name	Short Description
NO_EVENT	The system does not have the adequate information to send valid 'YES' or 'NO' states.
NO	The requested event is in 'NO' state.
YES	The requested event is in 'YES' state.
NOT_SUPPORTED	The requested data is not supported
FAULT	The module/sensor is currently faulted.

Related items:

BeltStatus – [section 14.2.28](#)
[GetVehicleData – section 11.8](#)
[OnVehicleData – section 11.9](#)
[Table of Contents](#)

14.1.40 IgnitionStableStatus

This enumeration describes the states of the ignition switch.

Element name	Short Description
IGNITION_SWITCH_NOT_STABLE	Ignition switch is not stable.
IGNITION_SWITCH_STABLE	Ignition switch is stable.
MISSING_FROM_TRANSMITTER	Either the data is not accessible or the sensor is broken.

Related items:

[BodyInformation – section 14.2.29](#)
[GetVehicleData – section 11.8](#)
[OnVehicleData – section 11.9](#)

[Table of Contents](#)

14.1.41 IgnitionStatus

This enumeration reflects the status of ignition.

Element name	Short Description
UNKNOWN	The information is not acceptable. The sensor cannot provide accurate data at this time.
OFF	The ignition is off.
ACCESSORY	The accessories are active (power windows, audio, display, etc.).
RUN	Ignition is active.
START	Starter is switched.
INVALID	The data is provided, but there is some sort of fault or problem.

Related items:

[BodyInformation – section 14.2.29](#)
[GetVehicleData – section 11.8](#)
[OnVehicleData – section 11.9](#)

[Table of Contents](#)

14.1.42 DeviceLevelStatus

The enumeration describes the possible status of the battery if reported.

Element name	Short Description
ZERO_LEVEL_BARS	The battery is low
ONE_LEVEL_BARS	The battery is in the first level bar.
TWO_LEVEL_BARS	The battery is in the second level bar
THREE_LEVEL_BARS	The battery is in the fourth level bar
FOUR_LEVEL_BARS	The battery is in the fifth level bar
NOT_PROVIDED	The data is not provided

Related items:

DeviceStatus – [section 14.2.30](#)

[GetVehicleData](#) – [section 11.8](#)

[OnVehicleData](#) – [section 11.9](#)

[Table of Contents](#)

14.1.43 Primary AudioSource

Reflects the current primary audio source (if selected).

Element name	Short Description
NO_SOURCE_SELECTED	No source for playing audio is selected.
USB	The audio is played from USB.
USB2	The audio is played from USB2
BLUETOOTH_STEREO_BTST	The audio in stereo format is played from Bluetooth.
LINE_IN	The audio is played from the line input.
IPOD	The audio is played from the IPod.
MOBILE_APP	The audio is played from the mobile application.

Related items:

DeviceStatus – [section 14.2.30](#)

[GetVehicleData](#) – [section 11.8](#)

[OnVehicleData](#) – [section 11.9](#)

[Table of Contents](#)

14.1.45 WiperStatus

This enumeration reflects the status of the wipers.

Element name	Short Description
OFF	The wipers are off.
AUTO_OFF	The wipers are automatically off after detecting the wipers do not need to be engaged (rain stopped, etc.).
OFF_MOVING	Means that though set to off, somehow the wipers have been engaged (physically moved enough to engage a wiping motion).
MAN_INT_OFF	The wipers are manually off after having been working.
MAN_INT_ON	The wipers are manually on.
MAN_LOW	The wipers are manually set to low speed.
MAN_HIGH	The wipers are manually set to high speed.
MAN_FLICK	The wipers are manually set for doing a flick.
WASH	The wipers are set to use the water from vehicle washer bottle for cleaning the windscreen.
AUTO_LOW	The wipers are automatically set to low speed.
AUTO_HIGH	The wipers are automatically set to high speed.
COURTESYWIPE	This is for when a user has just initiated a WASH and several seconds later a secondary wipe is automatically initiated to clear remaining fluid.
AUTO_ADJUST	This is set as the user moves between possible automatic wiper speeds.
STALLED	The wiper is stalled to its place. There may be an obstruction.
NO_DATA_EXISTS	The sensor / module cannot provide any information for wiper.

Related items:

[GetVehicleData – section 11.8](#)

[OnVehicleData – section 11.9](#)

[Table of Contents](#)

14.1.46 CompassDirection

The potential compass directions that IVI component may provide are listed below.

Element name	Short Description
NORTH	Represents the North compass direction
NORTHWEST	Represents the North-West compass direction
WEST	Represents the West compass direction

Element name	Short Description
SOUTHWEST	Represents the South-West compass direction
SOUTH	Represents the South compass direction
SOUTHEAST	Represents the South-East compass direction
EAST	Represents the East compass direction
NORTHEAST	Represents the North-East compass direction

Related items:

[GPSData – section 14.2.25](#)
[GetVehicleData – section 11.8](#)
[OnVehicleData – section 11.9](#)

[Table of Contents](#)

14.1.47 Dimension

The enumeration defines the supported dimensions of GPS.

Element name	Short Description
NO_FIX	No GPS at all
2D	Longitude and latitude
3D	Longitude and latitude and altitude

Related items:

[GPSData – section 14.2.25](#)
[GetVehicleData – section 11.8](#)
[OnVehicleData – section 11.9](#)

[Table of Contents](#)

14.1.48 VehicleDataresultCode

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

Element name	Short Description
SUCCESS	The requested data is accessible.
TRUNCATED_DATA	The data is truncated: not all of the requested information is available.
DISALLOWED	The request is not authorized in local policies.
USER_DISALLOWED	The request is included in a functional group explicitly blocked by the User.

Element name	Short Description
INVALID_ID	One of the provided IDs is not valid.
VEHICLE_DATA_NOT_AVAILABLE	The requested data is not available.
DATA_ALREADY_SUBSCRIBED	The requested data has been subscribed already.
DATA_NOT_SUBSCRIBED	The data is not subscribed.
IGNORED	The request is ignored because the intended result is already in effect.

Related items:

ReadDID – [section 11.3](#)

[DIDResult - section 14.2.24](#)

[Table of Contents](#)

14.1.49 TBTState

The enumeration describes the possible needs of HMI turn-by-turn.

Element name	Short Description
ROUTE_UPDATE_REQUEST	Request from HMI for updating the route.
ROUTE_ACCEPTED	Confirmation from HMI about accepting the route.
ROUTE_REFUSED	Information from HMI about the route refusal.
ROUTE_CANCELLED	Information from HMI about cancelling the route.
ETA_REQUEST	Request from HMI for Estimated time of arrival.
NEXT_TURN_REQUEST	Request from HMI for the information of the next turn.
ROUTE_STATUS_REQUEST	Request from HMI for the route status.
ROUTE_SUMMARY_REQUEST	Request from HMI for the route summary.
TRIP_STATUS_REQUEST	Request from HMI for the information about trip status.
ROUTE_UPDATE_REQUEST_TIMEOUT	Request from HMI for the timeout for waiting for the route updating.

Related items:

OnTBTClientState – [section 12.9](#)

[Table of Contents](#)

14.1.50 MethodName

Element name	Value	Short Description
ALERT	0	Defines the type of the mobile API request (Alert mobile API) which initiates some request on HMI e.g. (TTS.Speak, BC.PlayTone)

Element name	Value	Short Description
SPEAK	1	Defines the type of the mobile API request (Speak mobile request) which initiates some request on HMI (TTS.Speak, BC.PlayTone) as a part of itself
AUDIO_PASS_THROUGH	2	Defines the type of the mobile API request (PerformAudioPassThru mobile API) which initiates the request e.g. (TTS.Speak, BC.PlayTone) as a part of itself
ALERT_MANEUVER	3	Defines the type of the mobile API request (AlertMeneuver mobile API) which initiates the request on HMI e.g. (TTS.Speak, BC.PlayTone)

Related items:

PlayTone – [section 6.17](#)

Speak - [section 10.4](#)

[Table of Contents](#)

14.1.51 EmergencyState

Enumeration that describes possible states of "911 Assist" mode.

Element name	Short Description
EMERGENCY_ON	Emergency mode is ON on HU
EMERGENCY_OFF	Emergency mode is OFF on HU

Related items:

[Table of Contents](#)

14.1.52 TransportType

Element name	Value	Short Description
BLUETOOTH	0	
USB_IOS	1	
USB_AOA	2	
WIFI	3	

Related items:

UpdateAppList – [section 6.24](#)

OnAppRegistered – [section 6.13](#)

[Table of Contents](#)

14.2 Structures

14.2.1 HMIAplication

The structure defines the information about the application: its name, ID, the device concerned and etc. described in the table below.

Param Name	Type	Mandatory	Additional	Description
appName	String	true	Maxlength = 100	The mobile application name.
ngnMediaScreenAppName	String	false	Maxlength = 100	Provides an abbreviated version of the application name (may be displayed on the NGN media screen). If not provided, the appName should be used instead (and may be truncated if too long)
icon	String	false	-	Path to the application icon stored on HU.
deviceName	String	true	-	The name of the device where the identified application is running on.
appID	Integer	true	-	The application ID that remains unique during the ignition cycle. This ID will be sent by SDL further and must be provided by HMI within all the RPCs related to this application.
hmiDisplayLanguageDesired	Common.Language	false	-	The language that the application intends to use. See Language .
isMediaApplication	Boolean	false	-	Indicates whether the application is a media or a non-media one. Only media applications are allowed by SDL to stream audio to HU that is audible outside of the BT media source.
appType	Common.AppHMIType	false	array = true Minsize = 1 maxsize = 100	The HMI may use this information for determining what functionality should be available for the application (e.g. NAVIGATION type of application will require displaying the information and not playing the audio). See AppHMIType .
requestType	Common.RequestType	false	minsize=0 maxsize=100 array=true	The list of SystemRequest's RequestTypes allowed by policies for the named application (the app's SystemRequest sent with RequestType out of this list will get 'disallowed' response from SDL). If SDL sends an empty array – an

Param Name	Type	Mandatory	Additional	Description
				<p>any RequestType is allowed for this app.</p> <p>If SDL omits this parameter - none RequestType is allowed for this app</p> <p>(either this is a pre-registered app or such is dictated by policies).</p>

Related items:

UpdateAppList – [section 6.24](#)
 OnAppRegistered – [section 6.13](#)
 Language – [section 14.1.14](#)
 AppHMIType – [section 14.1.2](#)
[Table of Contents](#)

14.2.2 DeviceInfo

The structure is used for providing the information about the device connected to SDL:

Param Name	Type	Mandatory	Description
name	String	true	The name of the device.
id	String	true	The ID of the device connected. either hash of device's USB serial number (in case of USB connection) or hash of device's MAC address (in case of BlueTooth or WiFi connection). It remains unique between the ignition cycles for the same transport type .
transportType	Common.TransportType	false	The transport type the named-app's-device is connected over to HU (BlueTooth, USB or WiFi). Always returned by SDL in OnAppRegistered and UpdateAppList RPCs.
isSDLAllowed	Boolean	false	Sent by SDL in UpdateDeviceList. 'true' – if device is allowed for PolicyTable Exchange; 'false' – if device is NOT allowed for PolicyTable Exchange

Related items:

UpdateDeviceList – [section 6.1](#)
 OnAppRegistered – [section 6.13](#)
 UpdateAppList – [section 6.24](#)

AllowDeviceToConnect – [section 6.4](#)
 OnDeviceChosen – [section 6.9](#)
 OnFindApplications – [section 6.10](#)
 ActivateApp – [section 6.2](#)

[Table of Contents](#)

14.2.3 MenuParams

The structure is used for setting the position of the command/sub-menu and the name of the sub-menu to be added within the requests of `AddCommand` and `AddSubMenu`.

Param Name	Type	Mandatory	Additional	Description
parentID	Integer	false	minvalue = 0 maxvalue = 2000000000	<ul style="list-style-type: none"> - This value will NOT be provided for <code>AddSubMenu</code> request. - Unique ID of the sub menu, the command must be added to. - If not provided, the command must be added to the top-level application menu.
position	Integer	false	minvalue = 0 maxvalue = 1000	<p>This value is the position within the elements of top-level application menu where the command/sub-menu must be added to:</p> <ul style="list-style-type: none"> - If 0, the item must be inserted to the first position. - If 1, the item must be inserted to the second position. - Etc. <p>If the value is greater than or equal to the number of elements of the top-level application menu, the sub menu or a command must be appended to the end of the list.</p> <p>If omitted the entry must be added to the end of the list.</p>
menuName	String	true	maxlength = 500	The text that must be shown as a name of a command/sub -menu.

Related items:

`AddCommand` – for UI, [section 7.6](#)

`AddSubMenu` – for UI, [section 7.8](#)

[Table of Contents](#)

14.2.4 Choice

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

system during an application initiated interaction. For example, the application may request for the user's choice among several suggested ones: 'Yes', 'No', 'Skip'.

The given structure defines the UI representation of the choice.

The choices for VR interaction are added by the AddCommand ([section 9.4](#)), where cmdID param is provided for the choice identifier.

Param Name	Type	Mandatory	Additio nal	Description
choiceID	Integer	true	minvalue = 0 maxvalue = 65535	The unique within the concerned application identifier for this choice
menuName	String	false	Maxlength = 500	The text to be displayed in the onscreen menu indicating the name of the choice (e.g. 'Yes').
image	Common.Image	false	-	Image that must appear in the menu, representing this choice. See Image

Param Name	Type	Mandatory	Additional	Description
secondaryText	String	false	MaxLength = 500	Optional secondary text to display; e.g. address of POI in a search result entry
tertiaryText	String	false	MaxLength = 500	Optional tertiary text to display; e.g. distance to POI for a search result entry
secondaryImage	Common.Image	false	-	Optional secondary image struct for choice

Related items:

PerformInteraction – [section 7.10](#)

AddCommand – [section 7.6](#)

Image – [section 14.2.14](#)

[Table of Contents](#)

14.2.5 TimeFormat

The structure describes the format of time value to be set upon request from SDL with SetMediaClockTimer.

Param Name	Type	Mandatory	Additional	Description
------------	------	-----------	------------	-------------

Param Name	Type	Mandatory	Additional	Description
hours	Integer	true	minvalue = 0 maxvalue = 59	The hour of the media clock. Some display types only support a max of 19 hours. If out of range, the request must be rejected.
minutes	Integer	true	minvalue = 0 maxvalue = 59	The minute.
seconds	Integer	true	minvalue = 0 maxvalue = 59	The second.

Related items:

[SetMediaClockTimer – section 7.11](#)

[Table of Contents](#)

14.2.6 VrHelpItem

The structure provides the information about VR Help Menu item that must be displayed on UI representing the command of VR to the User.

Param Name	Type	Mandatory	Additional	Description
text	String	True	maxlength = 500	The text that must be displayed as a name for VR Help item.
image	Common.Image	False	-	Image that must be displayed by HU to represent the VR Help item. See Image
position	Integer	false	minvalue = 1 maxvalue = 100	This value is the position within the elements of VR Help menu where the item must be added to:

Related items:

[UI.PerformInteraction – section 7.10](#)

[UI.SetGlobalProperties – section 7.12](#)

[Image – section 14.2.14](#)

[Table of Contents](#)

14.2.7 DisplayCapabilities

The structure contains the information about the display capabilities.

Param Name	Type	Mandatory	Additional	Description
-------------------	-------------	------------------	-------------------	--------------------

Param Name	Type	Mandatory	Additional	Description
displayType	Common.DisplayType	true	-	The type of the display that is installed on HU. See DisplayType .
textFields	Common.TextFieldName	true	Array = true minsize = 0 maxsize = 100	A set of all fields that support displaying text data. If there are no textfields supported HMI must send the empty array. See TextFieldName .
mediaClockFormats	Common.MediaClockFormat	true	Array = true minsize = 1 maxsize = 100	A set of all supported formats of the media clock. See MediaClockFormat .
imageCapabilities	Common.ImageType	false	Array = true minsize = 0 maxsize = 2	The array of supported image types (static and/or dynamic). The empty array should be returned if the platform does not support displaying images. See ImageType .
graphicSupported	Boolean	true	-	The display's persistent screen supports referencing a static or dynamic image.

Related items:

DisplayType – [section 14.1.7](#)

MediaClockFormat – [section 14.1.8](#)

TextFieldName – [section 14.1.15](#)

ImageType – [section 5.1.18](#)

GetCapabilities – for UI, [section 7.2](#)

[Table of Contents](#)

14.2.8 TouchEventCapabilities

The structure contains the information about the display capabilities.

Param Name	Type	Mandatory	Description
pressAvailable	Boolean	true	
multiTouchAvailable	Boolean	true	

Param Name	Type	Mandatory	Description
doublePressAvailable	Boolean	true	

Related items:

GetCapabilities – for UI, [section 7.2](#)

[Table of Contents](#)

14.2.9 ImageResolution

The structure contains the information about the display capabilities.

Param Name	Type	Mandatory	Additional	Description
resolutionWidth	Integer	true	minvalue = 1 maxvalue = 10000	The image resolution width.
resolutionHeight	Integer	true	minvalue = 1 maxvalue = 10000	The image resolution height.

Related items:

GetCapabilities – for UI, [section 7.2](#)

[Table of Contents](#)

14.2.10 ScreenParams

The structure contains the information about the display capabilities.

Param Name	Type	Mandatory	Description
resolution	Common.ImageResolution	true	The resolution of the prescribed screen area.
touchEventAvailable	Common.TouchEventCapabilities	true	Types of screen touch events available in screen area.

Related items:

GetCapabilities – for UI, [section 7.2](#)

[Table of Contents](#)

14.2.11 ImageField

The structure contains the information about the display capabilities.

Param Name	Type	Mandatory	Additional	Description
name	Common.Image FieldName	true	-	The name that identifies the field. See ImageFieldName.
imageTypes supported	Common.FileT ype	false	Array = true minsize = 1 maxsize = 100	The image types that are supported in this field. See FileType.
imageResolu tion	Common.Image Resolution	false	-	The image resolution of this field

Related items:

GetCapabilities – for UI, [section 7.2](#)

[Table of Contents](#)

14.2.12 TextFieldStruct

The structure is used in requests for HMI for displaying the text message in the appropriate text area of the display.

Param Name	Type	Mandatory	Additional	Description
fieldName	Common.TextFie ldName	true	-	The name of the field where the text must be displayed in.
fieldText	String	true	Maxlengt h = 500	The text to be displayed.

Related items:

TextFieldName – [section 14.1.15](#)

Turn – [section 14.2.33](#)

Alert – [section 7.4](#)

Show – [section 7.5](#)

PerformInteraction – [section 7.10](#)

ScrollableMessage – [section 7.17](#)

PerformAudioPassThru – [section 7.18](#)

14.2.13 SoftButtonCapabilities

The structure contains the information about the soft buttons capabilities:

Param Name	Type	Mandatory	Description
shortPressAvailable	Boolean	true	Must be - 'true' if soft buttons support a short press - 'false' if not. See ButtonPress Mode for more information.
longPressAvailable	Boolean	true	Must be - 'true' if soft buttons support a LONG press - 'false' if not. See ButtonPress Mode for more information.
upDownAvailable	Boolean	true	Must be - 'true' if soft buttons support "button down" and "button up". - 'false' if not. See ButtonEvent Mode for more information.
imageSupported	Boolean	true	Must be - 'true' if soft buttons support referencing image - 'false' if not.

Related items:

[ButtonEventMode – section 14.1.18](#)

[ButtonPressMode – section 14.1.19](#)

[GetCapabilities – for Buttons, section 8.1](#)

[OnButtonEvent – section 8.2](#)

[OnButtonPress – section 8.3](#)

14.2.14 SoftButton

The structure describes all the values sent to HMI for displaying a soft button on UI.

Param Name	Type	Mandatory	Additional	Description
type	Common.SoftButtonType	true	-	Defines the type of the soft button: whether it must be displayed with a text, image or both text and image. See SoftButtonType .
text	String	false	maxlength = 500	If the value is present, UI must display the text on the corresponding soft button. This value is provided if the soft button type is defined as TEXT.
image	Common.Image	false	-	If the value is present, UI must display the image on the corresponding soft button. This parameter is provided if the soft button type is defined as IMAGE. See Image .
isHighlighted	Boolean	false	-	If ‘true’, the soft button must be highlighted. If ‘false’, must be not. If omitted, the button must NOT be highlighted.
softButton ID	Integer	true	minvalue = 0 maxvalue = 65535	The unique ID provided for identifying the soft button within the application. This value must be returned via OnButtonPress / OnButtonEvent as a customButtonId parameter.
systemAction	Common.SystemAction	false	-	Parameter that indicates whether clicking the SoftButton must call a specific system action. See SystemAction . If omitted the default system action should occur.

Related items:

SoftButtonType – [section 14.1.11](#)

SystemAction – [section 14.1.12](#)

Image – [section 14.2.14](#)

Alert – [section 7.4](#)
 Show – [section 7.5](#)
 ScrollableMessage – [section 7.17](#)
 OnButtonEvent – [section 8.2](#)
 OnButtonPress – [section 8.3](#)
 AlertManeuver – [section 12.2](#)
 ShowConstantTBT – [section 12.3](#)
 UpdateTurnList – [section 12.4](#)

[Table of Contents](#)

14.2.15 Image

The structure describes the type of and the path to the image to be displayed on UI upon corresponding request of SDL.

Param 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
imageType	Common.ImageType	true	-	Describes, whether it is a static or dynamic image.

Related items:

[ImageType – section 5.1.15](#)
[Choice – section 5.2.4](#)
[VrHelpItem – section 5.2.6](#)
[SoftButton – section 5.2.10](#)
[Turn – section 5.2.25](#)
[Show – section 7.1.5](#)
[AddCommand – section 7.1.6](#)
[SetAppIcon – section 7.1.15](#)
[ShowConstantTBT – section 12.3](#)

[Table of Contents](#)

14.2.16 ButtonCapabilities

The structure describes the hardware buttons capabilities.

Upon the request HMI must provide the list of the following information:

- The names of all existing/supported hardware buttons as described in the section 14.1.17.
- The availability of LONG/SHORT press for each existing/supported hardware button correspondingly
- The availability of UP/DOWN events for each existing/supported hardware button correspondingly

Param Name	Type	Mandatory	Description
name	Common.ButtonName	true	The name of supported/existing hardware buttons. See ButtonName
shortPressAvailable	Boolean	true	Must be 'true' if the button supports SHORT press mode. See ButtonPressMode
longPressAvailable	Boolean	true	The button supports LONG press mode. See ButtonPressMode
upDownAvailable	Boolean	true	The button supports "button down" and "button up". See ButtonEventMode

Related items:

ButtonName – [section 14.1.17](#)

ButtonEventMode – [section 14.1.18](#)

ButtonPressMode – [section 14.1.19](#)

GetCapabilities – for Buttons, [section 8.1](#)

[Table of Contents](#)

14.2.17 PresetBankCapabilities

Some HMI displays can duplicate hardware buttons with on-screen buttons. Such on-screen buttons have the same capabilities and functionality (e.g. play, pause, etc.) as hardware buttons (e.g. 'short press', 'long press').

HMI must provide the information about the availability of on-screen presets for supported hardware buttons.

Param Name	Type	Mandatory	Description
------------	------	-----------	-------------

onScreenPresetsAvailable	Boolean	true	Must be 'true' if on-screen custom presets are available for the hardware button of the provided ButtonName.
--------------------------	---------	------	--

Related items:

ButtonName – [section 14.1.17](#)

GetCapabilities – for Buttons, [section 8.1](#)

[Table of Contents](#)

14.2.18 AudioPassThruCapabilities

Describes different audio type configurations for PerformAudioPassThru.

e.g. 8kHz,8-bit,PCM

Param Name	Type	Mandatory	Description
samplingRate	Common.SamplingRate	true	
bitsPerSample	Common.BitsPerSample	true	
audioType	Common.AudioType	true	

Related items:

GetCapabilities – for UI, [section 7.2](#)

[Table of Contents](#)

14.2.19 TouchCoord

Defines User's touch coordinates

Param Name	Type	Mandatory	Description
------------	------	-----------	-------------

x	Integer	True minvalue="0" maxvalue="10000"	The x coordinate of the touch.
y	Integer	True minvalue="0" maxvalue="10000"	The y coordinate of the touch.

Related items:

OnTouchEvent – [section 7.25](#)

[Table of Contents](#)

14.2.20 TouchEvent

Defines the user's touch in a moment of time

Param Name	Type	Mandatory	Additional	Description
id	Integer	true	minvalue = 0 maxvalue = 9	Finger / stylus ID
ts	Integer	true	minvalue="0" maxvalue="2147483647" minszie="1" maxsize="1000"	<p>The time that the touch was recorded. This number can be time since the beginning of the session or something else as long as the units are in milliseconds.</p> <p>The timestamp is used to determine the rate of change in position of a touch.</p> <p>The application also uses this time to verify whether two touches, with different ids, are part of a single action by the user.</p> <p>If there is only a single timestamp in this array, it is the same for every coordinate in the coordinates array.</p>
c	Common.TouchCoord	true	array="true" minsize="1" maxsize="1000"	Describe the ellipse approximating the input shape

Related items:

OnTouchEvent – [section 7.25](#)

[Table of Contents](#)

14.2.21 KeyboardProperties

Configuration of on-screen keyboard (if available).

Param Name	Type	Mandatory	Additional	Description

language	Common.Language	false		The keyboard language.
keyboardLayout	Common.KeyboardLayout	false		Desired keyboard layout.
sendDynamicEntry	Boolean	false		In this mode, all keypresses will be sent as they occur. If disabled, entire string of text will be returned only once submitted by user. If omitted, this value will be set to FALSE.
limitedCharacterList	String	false	Array = true maxlength = 1 minsize = 1 maxsize = 100	Array of keyboard characters to enable. All omitted characters will be greyed out (disabled) on the keyboard. If omitted, the entire keyboard will be enabled.
autoCompleteText	String	false	maxlength = 1000	Allows an app to prepopulate the text field with a suggested or completed entry as the user types

Related items:

[UI.SetGlobalProperties - section 7.12](#)

[Table of Contents](#)

14.2.22 TTSCheck

The following structure defines a TTS chunk that consists of the text/phonemes to be spoken by the TTS module of HU:

Param Name	Type	Mandatory	Maxlength	Description
text	String	true	500	The text or phonemes to speak
type	Common.SpeechCapabilities	true	-	Describes, whether it is text or a specific phoneme set. See SpeechCapabilities.

Related items:

[Speak – section 10.4](#)

[SetGlobalProperties – for TTS, section 10.8](#)

[Table of Contents](#)

14.2.23 VehicleType

The following structure describes the information about vehicle type: its manufacturer, model, model year and trim.

Param Name	Type	Mandatory	Additional	Description
make	String	false	maxlength = 500	Make of the vehicle (e.g. Ford)
model	String	false	maxlength = 500	Model of the vehicle (e.g. Fiesta)
modelYear	String	false	maxlength = 500	Model Year of the vehicle (e.g. 2013)
trim	String	false	maxlength = 500	Trim of the vehicle (e.g. SE)

Related items:

GetVehicleType – [section 11.2](#)

[Table of Contents](#)

14.2.24 DIDResult

The structure describes the information returned for the ReadDID request.

Param Name	Type	Mandatory	Additional	Description
resultCode	Common.VehicleDataResultCode	true	–	Individual DID result code (see VehicleDataResultCode).
didLocation	Integer	true	minvalue = 0 maxvalue = 65535	The address of DID location from the ReadDID request.
data	String	false	maxlength = 5000	The DID data which is the hex byte string of however many bytes are stored at that location

Related items:

VehicleDataResultCode – [section 14.1.48](#)

ReadDID – [section 11.3](#)

[Table of Contents](#)

14.2.25 GPSData

The following structure describes the GPS data, which IVI module is expected to return for the request of SDL.

Param Name	Type	Mandatory	Maxvalue	Description
longitudeDegrees	Float	true	minvalue = -180 maxvalue = 180	Position longitude in degrees

Param Name	Type	Mandatory	Maxvalue	Description
latitudeDegrees	Float	true	minvalue = -90 maxvalue = 90	Position latitude in degrees
utcYear	Integer	false	minvalue = 2010 maxvalue = 2100	The current UTC year.
utcMonth	Integer	false	minvalue = 1 maxvalue = 12	The current UTC month.
utcDay	Integer	false	minvalue = 1 maxvalue = 31	The current UTC day
utcHours	Integer	false	minvalue = 0 maxvalue = 23	The current UTC hour.
utcMinutes	Integer	false	minvalue = 0 maxvalue = 59	The current UTC minute
utcSeconds	Integer	false	minvalue = 0 maxvalue = 59	The current UTC second
compassDirection	Common.CompassDirection	false	-	The information about compass direction (see section 5.1.25).
pdop	Float	false	minvalue = 0 maxvalue = 10	Positional Dilution Of Precision
hdop	Float	false	minvalue = 0 maxvalue = 10	Horizontal Dilution Of Precision
vdop	Float	false	minvalue = 0 maxvalue = 10	Vertical Dilution Of Precision
actual	Boolean	false	-	The information about actuality of GPS data should be returned: - 'true', if actual - 'false', if inferred
satellites	Integer	false	minvalue = 0 maxvalue = 31	Number of satellites in view
dimension	Common.Dimension	false	-	The supported GPS dimension (see section 14.1.47)
altitude	Float	false	minvalue = -10000 maxvalue = 10000	Altitude in meters
heading	Float	false	minvalue = 0 maxvalue = 359.99	'0' is considered as heading North. Resolution accepted by SDL is 0.01.
speed	Float	false	minvalue = 0 maxvalue = 500	The speed in KPH

Related items:

CompassDirection – [section 14.1.46](#)

Dimension – [section 14.1.47](#)

GetVehicleData – [section 11.8](#)

OnVehicleData – [section 11.9](#)

[Table of Contents](#)

14.2.26 TireStatus

The following structure describes the status and pressure of vehicle tires.

Param Name	Type	Mandatory	Description
pressureTelltale	Common.WarningLightStatus	false	Status of the Tire Pressure Telltale. See WarningLightStatus.
leftFront	Common.SingleTireStatus	false	The status of the left front tire.
rightFront	Common.SingleTireStatus	false	The status of the right front tire.
leftRear	Common.SingleTireStatus	false	The status of the left rear tire.
rightRear	Common.SingleTireStatus	false	The status of the right rear tire.
innerLeftRear	Common.SingleTireStatus	false	The status of the inner left rear.
innerRightRear	Common.SingleTireStatus	false	The status of the inner right rear.

Related items:

WarningLightStatus – [section 14.1.38](#)

SingleTireStatus – [section 14.2.27](#)

GetVehicleData – [section 11.8](#)

OnVehicleData – [section 11.9](#)

[Table of Contents](#)

14.2.27 SingleTireStatus

The structure provides the status of component volume.

Param Name	Type	Mandatory	Description
status	Common.ComponentVolumeStatus	true	See ComponentVolumeStatus

Related items:

ComponentVolumeStatus – [section 14.1.36](#)

GetVehicleData – [section 11.8](#)

OnVehicleData – [section 11.9](#)

[Table of Contents](#)

14.2.28 BeltStatus

The structure references the signals from the sensors that detect whether the seat belt is deployed or buckled.

Param Name	Type	Mandatory	Description
driverBeltDeployed	Common.VehicleDataEventStatus	false	The driver seat belt is deployed.
passengerBeltDeployed	Common.VehicleDataEventStatus	false	The passenger seat belt is deployed.
passengerBuckleBelted	Common.VehicleDataEventStatus	false	The passenger seat belt is buckled.
driverBuckleBelted	Common.VehicleDataEventStatus	false	The driver seat belt is buckled.
leftRow2BuckleBelted	Common.VehicleDataEventStatus	false	The left seat belt of the 2 nd row is buckled.
passengerChildDetected	Common.VehicleDataEventStatus	false	The child passenger is detected.
rightRow2BuckleBelted	Common.VehicleDataEventStatus	false	The right seat belt of the 2 nd row is buckled.
middleRow2BuckleBelted	Common.VehicleDataEventStatus	false	The middle seat belt of the 2 nd row is buckled.
middleRow3BuckleBelted	Common.VehicleDataEventStatus	false	The middle seat belt of the 3 rd row is buckled.
leftRow3BuckleBelted	Common.VehicleDataEventStatus	false	The left seat belt of the 3 rd row is buckled.
rightRow3BuckleBelted	Common.VehicleDataEventStatus	false	The right seat belt of the 3 rd row is buckled.
leftRearInflatableBelted	Common.VehicleDataEventStatus	false	The left rear inflatable is belted.
rightRearInflatableBelted	Common.VehicleDataEventStatus	false	The right rear inflatable is belted.
middleRow1BeltDeployed	Common.VehicleDataEventStatus	false	The seat belt of the middle row is deployed.
middleRow1BuckleBelted	Common.VehicleDataEventStatus	false	The seat belt of the middle row is buckled.

Related items:

VehicleDataEventStatus – [section 14.1.39](#)

GetVehicleData – [section 11.8](#)

OnVehicleData – [section 11.9](#)

[Table of Contents](#)

14.2.29 BodyInformation

The structure defines the information about the park brake and ignition.

Param Name	Type	Mandatory	Description
------------	------	-----------	-------------

Param Name	Type	Mandatory	Description
parkBrakeActive	Boolean	true	The information about the park brake: - 'true', if active - 'false' if not.
ignitionStableStatus	Common.IgnitionStableStatus	true	The information about stability of the ignition switch. See IgnitionStableStatus.
ignitionStatus	Common.IgnitionStatus	true	The information about ignition status. See IgnitionStatus.
driverDoorAjar	Boolean	false	
passengerDoorAjar	Boolean	false	
rearLeftDoorAjar	Boolean	false	
rearRightDoorAjar	Boolean	false	

Related items:

IgnitionStableStatus – [section 14.1.40](#)

IgnitionStatus – [section 14.1.41](#)

GetVehicleData – [section 11.8](#)

OnVehicleData – [section 11.9](#)

[Table of Contents](#)

14.2.30 DeviceStatus

The structure reflects the information about the Head Unit status.

Param Name	Type	Mandatory	Description
voiceRecOn	Boolean	false	Must be 'true' if the voice recording is on.
btIconOn	Boolean	false	Must be 'true' if Bluetooth icon is displayed.
callActive	Boolean	false	Must be 'true' if there is an active call.
phoneRoaming	Boolean	false	Must be 'true' if there is a phone roaming.
textMsgAvailable	Boolean	false	Must be 'true' if the text message is available.
battLevelStatus	Common.DeviceLevelStatus	false	Device battery level status. See DeviceLevelStatus.
stereoAudioOutputMuted	Boolean	false	Must be 'true' if stereo audio output is muted.

Param Name	Type	Mandatory	Description
monoAudioOutputMuted	Boolean	false	Must be ‘true’ if mono audio output is muted.
signalLevelStatus	Common.DeviceLevelStatus	false	Device signal level status. See DeviceLevelStatus.
primary AudioSource	Common.PrimaryAudioSource	false	Primary audio source. See Primary AudioSource.
eCallEventActive	Boolean	false	Must be ‘true’ if emergency call event is active.

Related items:

DeviceLevelStatus – [section 14.1.42](#)
 Primary AudioSource – [section 14.1.43](#)
 GetVehicleData – [section 11.8](#)
 OnVehicleData – [section 11.9](#)

[Table of Contents](#)

14.2.31 HeadLampStatus

The structure describes the status of the headlights.

Param Name	Type	Mandatory	Description
lowBeamsOn	Boolean	true	Status of the low beam lamps.
highBeamsOn	Boolean	true	Status of the high beam lamps.
ambientLightSensorStatus	Common.AmbientLightStatus	true	Status of the ambient light sensor.

Related items:

GetVehicleData – [section 11.8](#)
 OnVehicleData – [section 11.9](#)

[Table of Contents](#)

14.2.32 Turn

The structure represents the information for TBT navigation.

Param Name	Type	Mandatory	Description
navigationText	Common.TextFieldStruct	false	Contains the information text and the field for the text to be displayed in. Uses navigationText from TextFieldName
turnIcon	Common.Image	false	The image that represents the information about the turn. See Image

Related items:

UpdateTurnList – [section 12.4](#)
 TextFieldStruct – [section 14.2.11](#)
 TextFieldName – [section 14.1.15](#)
 Image – [section 14.2.14](#)

[Table of Contents](#)

14.2.33 HMICapabilities

Param Name	Type	Mandatory	Additional	Description
navigation	boolean	false		Availability of build in Navigation. True: available, False: not available
phoneCall	boolean	false		Availability of build in phone. True: available, False: not available

Related items:

UI.GetCapabilities – [section 7.2](#)

[Table of Contents](#)

15 Other

15.1 SDL's configuration file structure (ini-file)

15.1.1 Description

To configure some specific SDL rules or to define the filepaths and other other SDL settings, *smartDeviceLink.ini* file is used. The file is divided into a sections, each section relates to the configuration of some functional area. Some of the settings have no reference to HMI behavior, anyway they are described for information purposes to understand particular properties. See [15.3.2 Structure](#) for getting detailed information about each group.

15.1.2 Structure

15.1.2.1 HMI

Parameter	Type	Example	Description
LaunchHMI	Boolean	LaunchHMI = false	Defines if SDL will start HMI process or will be running without HMI initiation(for open-source project) True - HMI will be started by SDL False - SDL will be running without HMI starting

Parameter	Type	Example	Description
ServerAddress	String	ServerAddress = 127.0.0.1	Address to connect HMI (web-socket only) See also WebSocket Transport
ServerPort	Integer	ServerPort = 8087	ServerPort (web-socket only) See also WebSocket Transport
VideoStreaming Port	Integer	VideoStreamingPort = 5050	Port number for streaming video (in case of socket type) See also Configuring audio/video streaming parameters in smartDeviceLink.ini file and 12.5 StartStream
AudioStreaming Port	Integer	AudioStreamingPort = 5080	Port number for streaming audio (in case of socket type) See also Configuring audio/video streaming parameters in smartDeviceLink.ini file and 12.7 StartAudioStream
LinkToWebHMI	String	LinkToWebHMI = "HMI/index.html"	Default HMI page to be run

15.1.2.2 MAIN

Parameter	Type	Example	Description
SDLVersion	String	SDLVersion =	SDL version to be sent in RegisterAppInterface response, if empty - an empty parameter will be returned to mobile app
LogsEnabled	Boolean		SDL logging output enabled/disabled on system
AppConfigFolder	String	AppConfigFolder =	The path to application configuration folder where hmi_capabilities, smartDeviceLink.ini, log4cxx.properties are stored. Storage of Policy table files (preload, snapshot) is valid for OpenSource only
AppResourceFolder		AppResourceFolder =	Contains application resources, e.g. audio8bit.wav to which audio path thru data is written from the microphone for further transferring to mobile app
AppStorageFolder	String	AppStorageFolder = storage	The root folder for storing output files, e.g. .wav

Parameter	Type	Example	Description
ThreadStackSize	Integer	ThreadStackSize = 20480	ThreadStackSize used by SDL only if its required by system/platform, wisean empty value is defined or no such parameter exists, stack size will be PTHREAD_STACK_MIN(only SDL's configurable value), which for Ubuntu: THREAD_STACK_MIN = 16384 QNX: PTHREAD_STACK_MIN = 256
MixingAudioSupported	Boolean	MixingAudioSupported = true	Defines if HMI support attenuated mode (able to mix audio sources)
HMICapabilities	String	HMICapabilities = hmi_capabilities.json	The filepath to the file of default HMICapabilities. In case HMI doesn't send some capabilities to SDL, the values from the file are used by SDL
MaxCmdID	Int64	MaxCmdID = 2000000000	Maximum cmdId of VR command which may be registered on SDL
HMIHeartBeatTimeout	Integer	HMIHeartBeatTimeout = 3000;	HMI's heartbeat timeout. The value specifies seconds to send heartbeat to HMI from SDL
DefaultTimeout	Integer	DefaultTimeout = 20000	The timeout the response must be send to mobile application (SDL must respond after this timeout if HMI haven't respond on a request)
AppDirectoryQuota	Int64	AppDirectoryQuota = 104857600	Definines folder size in bytes limitation for any application on SDL
AppHMILevelNoneTimeScaleMaxRequests	Integer	AppHMILevelNoneTimeScaleMaxRequests = 0	Number of the requests allowed for the application in NONE HMI Level. If exceeded, the application will be unregistered
AppHMILevelNoneRequestsTimeScale	Integer	AppHMILevelNoneRequestsTimeScale = 10	Time period in which AppHMIlevelNoneTimeScaleMaxRequests requests number allowed for the application in NONE HMI Level. If exceeded, the application will be unregistered
AppTimeScaleMaxRequests	Integer	AppTimeScaleMaxRequests = 0	Number of the requests allowed for the application in any HMI Level except of NONE. If exceeded, the application will be unregistered

Parameter	Type	Example	Description
AppRequestsTimeScale	Integer	AppRequestsTimeScale = 10	Time period in which AppTimeScaleMaxRequests requests number allowed for the application in any HMI Level except of NONE. If exceeded, the application will be unregistered
PendingRequestsAmount	Integer	PendingRequestsAmount = 0	Number of the SDL pending requests allowed for an application. If exceeded, the application will be unregistered
HeartBeatTimeout	Integer	HeartBeatTimeout = 0	Heart beat timeout used for protocol v3. Timeout must be specified in milliseconds. If timeout is 0 heart beat between Mobile device and SDL will be disabled. SDL sends heartbeat and waiting for a response within HeartBeatTimeout timeframes, otherwise it consider connection to be broken.
SupportedDiagModes	String Array=true	SupportedDiagModes = 0x01, 0x02, 0x03, 0x05, 0x06, 0x07, 0x09, 0x0A, 0x18, 0x19, 0x22, 0x3E	The list of diagnostic modes supported on a vehicle. Only the stated values are allowed by SDL in terms of DiagnosticMessage RPC, others are rejected
SystemFilePath	String	SystemFilePath = /fs/images/ivsu_cache	The path to the system file directory for interoperation between SDL and System (e.g. IVSU files and others). If parameter is empty, SDL uses /tmp/fs/mp/images/ivsu_cache by default
TimeTestingPort	Integer	TimeTestingPort = 8090	Port to obtain the performance information about messages processing on different component levels. Enabled if SDL built with TIME_TESTER flag
ReadDIDRequest	Integer, Array[2] =true	ReadDIDRequest = 5, 1	Limitation for a number of ReadDID requests (the 1 st value) per seconds (the 2 nd value)
GetVehicleDataRequest	Integer, Array[2] =true	GetVehicleDataRequest = 5, 1	Limitation for a number of GetVehicleData requests (the 1 st value) per seconds (the 2 nd value)

15.1.2.4 MEDIA MANAGER

Parameter	Type	Example	Description
-----------	------	---------	-------------

Parameter	Type	Example	Description
StartStreamRetry	Integer, Array[2] =true	StartStreamRetry = 3, 1000	Where the 1 st one of the values is a number of retries and the 2 nd number is a timeout in seconds for request frequency
EnableRedecoding	Boolean	EnableRedecoding = false	If decoding on HMI is required?APTHR
NamedVideoPipePath	String	NamedVideoPipePath = video_stream_pipe	Named pipe path will be constructed using AppStorageFolder + NamedVideoPipePath Named pipe will be defined as: 1) <fileName> file in AppStorageFolder if the format of the parameter matchs the template: NamedVideoPipePath = <fileName> 2) <fileName> file in <path> folder if the format of the parameter matchs the template: NamedVideoPipePath = <path/><fileName> 3) "video_stream_pipe" file in AppStorageFolder if NamedVideoPipePath has an empty value "video_stream_pipe" file in AppStorageFolder if SDL has no permissions to write to <path/><fileName> defined in NamedVideoPipePath
NamedAudioPipePath	String	NamedAudioPipePath = audio_stream_pipe	Named pipe path will be constructed using AppStorageFolder + NamedAudioPipePath Named pipe will be defined as: 1) <fileName> file in AppStorageFolder if the format of the parameter matchs the template: NamedAudioPipePath = <fileName> 2) <fileName> file in <path> folder if the format of the parameter matchs the template: NamedAudioPipePath = <path/><fileName> 3) "audio_stream_pipe" file

Parameter	Type	Example	Description
			in AppStorageFolder if NamedAudioPipePath has an empty value "audio_stream_pipe" file in AppStorageFolder if SDL has no permissions to write to <path/file Name> defined in NamedAudioPipePath
VideoStreamFile	String	VideoStreamFil e = video_stream_f ile	4) File path will be constructed using AppStorageFolder + VideoStreamFile
AudioStreamFile	String	AudioStreamFil e = audio_stream_f ile	4) File path will be constructed using AppStorageFolder + name
RecordingFileS ource	String	RecordingFileS ource = audio.8bit.wav	Recording file source (used for audio pass thru emulation only), not applicable for SYNC HU system
StopStreaming Timeout	Integer	StopStreamingT imeout = 1000	The timeout in milliseconds for mobile to stop streaming or end up session
RecordingFileNa me	String	RecordingFileN ame = audio.wav	The filename of the file for audio pass thru data recording
MQAudioPath	String	MQAudioPath = /dev/mqueue/AP pLinkAudioPass	The name of MQ which is used by system HMI to send AudioPassThru data
AudioDat aStopped Timeout	Integer	AudioDataStopp edTimeout = 1000	Defines time in milliseconds for SDL to wait for the next package of raw data over audio service.

15.1.2.5 GLOBAL PROPERTIES

Parameter	Type	Example	Description
TTSDelimiter	Char	TTSDelimiter = ,	Defines the delimiter, which will be appended to each TTS chunk, e.g. helpPrompt/timeoutPr ompt

Parameter	Type	Example	Description
HelpPromt	String	HelpPromt = Please speak one of the following commands,Please say a command	Default prompt items, separated by comma. The value is set up in case ResetGlobalProperties request from the application or SDL (in case of TTSGlobalPropertiesTimeout)
TimeOutPromt	String	TimeOutPromt = Please speak one of the following commands,Please say a command	Default prompt to be spoken by VR-engine timeout (when system expects some action by user and the time for this action is going to be out)
HelpTitle	String	HelpTitle = Available Vr Commands List	Text to be set-up for default value of the title of help list
TTSGlobalPropertiesTimeout	Integer	TTSGlobalPropertiesTimeout = 20	In case mobile app didn't send global properties request, then default global properties will be sent defined timeout in seconds Max value TTSGlobalPropertiesTimeout 64K

15.1.2.6 FILESYSTEM RESTRICTIONS

Parameter	Type	Example	Description
PutFileRequest	Integer	PutFileRequest = 5	Max allowed number of PutFile requests for one application in NONE. The application will be unregistered in case the number of requests exceeded the limitation
DeleteFileRequest	Integer	DeleteFileRequest = 5	Max allowed number of DeleteFile requests for one application in NONE. The application

Parameter	Type	Example	Description
ListFilesRequest	Integer	ListFilesRequest = 5	will be unregistered in case the number of requests exceeded the limitation Max allowed number of ListFiles requests for one application in NONE. The application will be unregistered in case the number of requests exceeded the limitation

15.1.2.6 VR COMMANDS

Parameter	Type	Example	Description
HelpCommand	String	HelpCommand = Help	Default command which initiates VR Help of the application

15.1.2.7 ApplInfo

Parameter	Type	Example	Description
AppInfoStorage	String	AppInfoStorage = app_info.dat	The path for applications information storage (for resumption purposes)

15.1.2.8 Policy

Parameter	Type	Example	Description
EnablePolicy	Boolean	EnablePolicy = true	Turn on/off policies
PreloadedPT	String	PreloadedPT = sdl_preloaded_pt.json	The filename of the preloaded policy table

Parameter	Type	Example	Description
PathToSnapshot	String	PathToSnapshot = sdl_snapshot.json	The filename of the policy table snapshot
AttemptsToOpenPolicyDB	Integer	AttemptsToOpenPolicyDB = 10	Number of attempts to open policy DB
OpenAttemptTimeoutMs	Integer	OpenAttemptTimeoutMs = 1000	Timeout between attempts during opening DB in milliseconds

15.1.2.9 TransportManager

Parameter	Type	Example	Description
TCPAdapterPort	Integer	TCPAdapterPort = 12345	Listening port for incoming mobile connection
MMEDatabase	String	MMEDatabase = /dev/qdb/mEDIASERVICE_db	DB which contains information about IAP
EventMQ	String	EventMQ = /dev/mqueue/ToSDLCoreUSBAdapter	MQ for connection/disconnection events
AckMQ	String	AckMQ = /dev/mqueue/FromSDLCoreUSBAdapter	MQ for acknowledgement of connection/disconnection

15.1.2.10 ProtocolHandler

Parameter	Type	Example	Description
MaximumPayloadSize	Integer	MaximumPayloadSize = 1488	Packet with a payload bigger than # MaximumPayloadSize will be marked as a malformed. Parameter must be used for protocol v3 or higher. 1) In case of empty value, SDL uses the default value of 1488 bytes for validating MTU (maximum transferring unit) size. 2) If the value defined in MaximumPayloadSize parameter is less

Parameter	Type	Example	Description
			than 1488, the default value of 1488 bytes must be used by SDL.
FrequencyCount	Integer	FrequencyCount = 1000	Application shall send not more than #FrequencyCount messages per #FrequencyTime mSecs Frequency check could be disabled by setting #FrequencyTime or #FrequencyCount to "0"
FrequencyTime	Integer	FrequencyTime = 1000	The limitation in ms for sending number of the requests not more than #FrequencyCount
MalformedMessageFiltering	Integer	MalformedMessageFiltering = true	Enable filtering transport data stream; On #MalformedMessageFiltering disabling, SDL will close the connection with the first malformed message detection
MalformedFrequencyCount	Integer	MalformedFrequencyCount = 10	Boundary value of malformed message detection for connection closure Can be disabled by setting #MalformedFrequencyTime or #MalformedFrequencyCount to "0"
MalformedFrequencyTime	Integer	MalformedFrequencyTime = 1000	Boundary value of malformed message detection for connection closure. In case the frequency of malformed requests will be more than per #MalformedFrequencyTime, SDL will close the connection

15.1.2.11 Security Manager

Parameter	Type	Example	Description
-----------	------	---------	-------------

Parameter	Type	Example	Description
Protocol	String	Protocol = TLSv1.2	Protocol version
KeyPath	String	KeyPath = client.key	Certificate and key path to pem file
CertificatePath	String	CertificatePath = client.crt	Certificate and key path to pem file
SSLMode	String	SSLMode = CLIENT	SSL mode could be SERVER or CLIENT
CipherList	String	CipherList = ALL	Could be "ALL" ciphers or a list of chosen
VerifyPeer	Boolean	VerifyPeer = true	Defines if Mobile app certificate must be verified or not (could be used in both SSLMode Server and Client)
CACertificatePath	String	CACertificatePath = .	Preloaded CA certificates directory
ForceProtectedService	String	ForceProtectedService = Non	Force protected services (could be id's from 0x01 to 0xFF or "Non" value)
ForceUnprotectedService	String	ForceUnprotectedService = Non	Force unprotected services (could be id's from 0x01 to 0xFF or "Non" value)

15.1.2.12 ApplicationManager

Parameter	Type	Example	Description
ApplicationListUpdateTimeout	Integer	ApplicationListUpdateTimeo ut = 2000	Application list update timeout ms
ThreadPoolSize	Integer	ThreadPoolSize = 1	Max allowed threads for handling mobile requests. Currently max allowed is 2

Parameter	Type	Example	Description
HashStringSize	Integer	HashStringSize = 32	The max size of hash which is used by OnHashUpdated()

15.1.2.13 SDL4

Parameter	Type	Example	Description
EnableProtocol4	Boolean	EnableProtocol4 = true	Enables SDL 4.0 support
AppIconsFolder	String	AppIconsFolder = /fs/rwdata/storage/sdl	Path where apps icons must be stored
AppIconsFolderMaxSize	Integer	AppIconsFolderMaxSize = 104857600	Max size of the folder in bytes
AppIconsAmountToRemove	Integer	AppIconsAmountToRemove = 1	Amount of oldest icons to remove in case of max folder size was reached

15.1.2.14 Resumption

Parameter	Type	Example	Description
ApplicationResumingTimeout	Integer	ApplicationResumingTimeout = 3000	# Timeout in milliseconds for resumption Application HMI_Level and resolving conflicts in case if multiple applications initiate resumption
AppSavePersistentDataTimeout	Integer	AppSavePersistentDataTimeout = 10000	Timeout in milliseconds for periodic saving resumption persistent data
ResumptionDelayBeforeIgn	Integer	ResumptionDelayBeforeIgn = 30	Timeout in seconds to store hmi_level for media app before ign_off
ResumptionDelayAfterIgn	Integer	ResumptionDelayAfterIgn = 30	Timeout in seconds to restore hmi_level for media app after sdl run

Parameter	Type	Example	Description
UseDBForResumption	Boolean	UseDBForResumption = false	Resumption ctrl uses JSON if UseDBForResumption =false for store data otherwise uses DB
AttemptsToOpenResumptionDB	Integer	AttemptsToOpenResumptionDB = 10	Number of attempts to open resumption DB
OpenAttemptTimeoutMsResumptionDB	Integer	OpenAttemptTimeoutMsResumptionDB = 1000	Timeout between attempts during opening DB in milliseconds

15.2 HMI<->SDL fake parameters processing

15.2.1 Description

SDL may receive fake parameters in requests/responses/notifications coming from HMI. Because of this, several rules are applied on SDL to manage such kind of messages received. The rules are applied to all HMI API, defined between SDL and HMI.

SDL Rules:

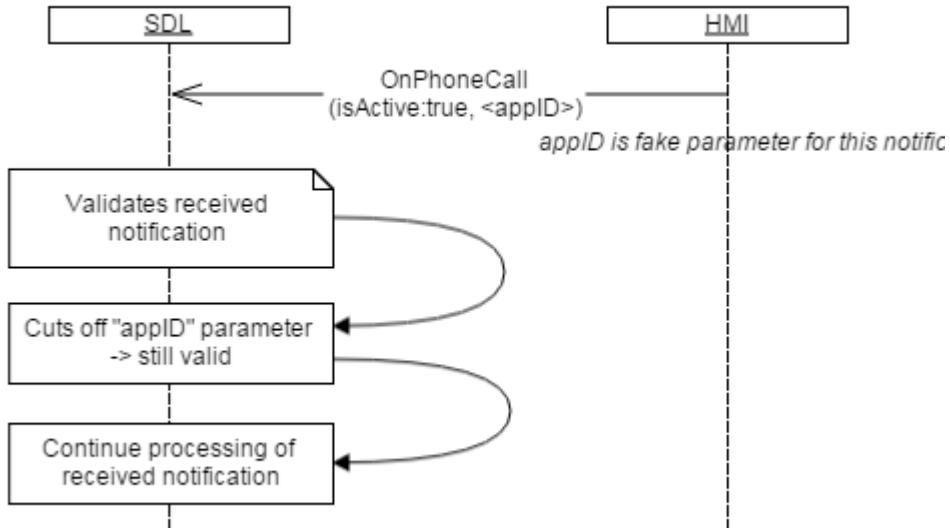
- 1) SDL must cut off all fake parameters coming from HMI and operate according to the requirements with the valid data left. See [15.2.2.1 Fake parameters cut off and the valid data left](#)

- 2) If the request/response/notification became invalid after fake parameters cut off, SDL must:
 - a) Notify an application about the appropriate mobile request/response (see [15.2.2.3 Fake parameters cut off and invalid data left related to mobile API](#))

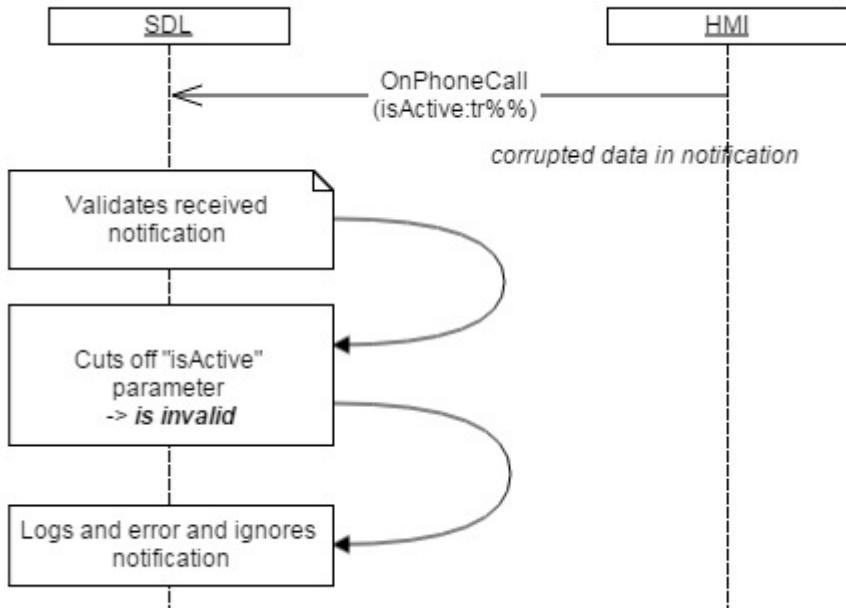
 - b) Ignore the request/response if it doesn't relate to the app's mobile API execution and is used just for interoperation between SDL and HMI (see [15.2.2.2 Fake parameters cut off and invalid data left \(no mobile API relationship\)](#))

15.2.2 Sequence diagrams

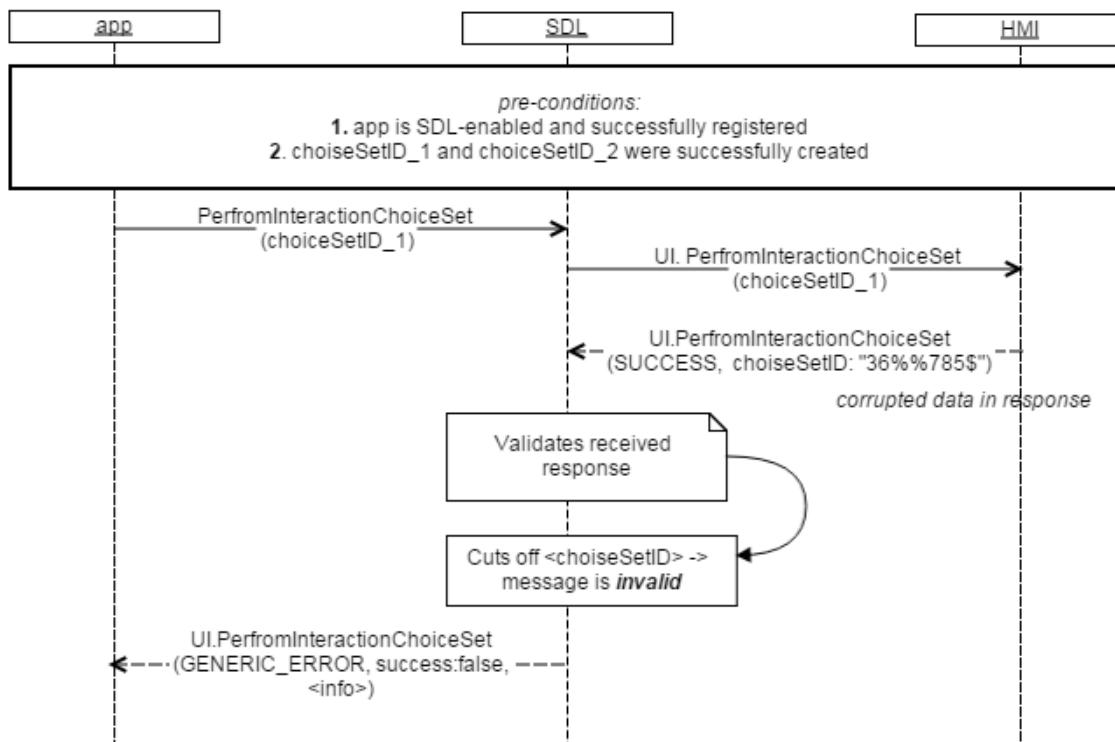
15.2.2.1 Fake parameters cut off and the valid data left



15.2.2.2 Fake parameters cut off and invalid data left (no mobile API relationship)



15.2.2.3 Fake parameters cut off and invalid data left related to mobile API



15.3 HMI Capabilities JSON file

15.3.1 Description

When SDL doesn't get capabilities from HMI (by any reason via any of HMI API: UI/TTS/VR/Buttons.GetCapabilities), it must use hmi_capabilities.json file's data to set-up default values for operation with HMI. The file specifies applicable HMI capabilities which allowSDL to get internal information which capabilities are supported and how the data may be processed by HMI components (e.g. supported data fields and events, settings etc).

Hmi_capabilities.json file's path is defined

15.3.2 Sections

15.3.2.1 UI

To apply UI capabilities the data received in a response from [UI.GetCapabilities](#) is used. In case no answer from HMI or unavailability to get this data, the default UI capabilities will be applied on SDL.

Param Name	Elements	Description	Reference
------------	----------	-------------	-----------

Language		Current VR language	To define all UI supported capabilities parameters see an example of hmi_capabilities.json section TTS
languages		Languages supported by VR engine	
displayCapabilities	displayType keyboardProperties textFields imageFields mediaClockFormats graphicSupported imageCapabilities	<p>displayType: the type of the display on HMI</p> <p>keyboardProperties: values applied on SDL and HMI in case of a default UI.SetGlobalProperties initiated by SDL (if no SetGlobalProperties are sent from mobile application on registering)</p> <p>textFields: text fields supported on HMI</p> <p>imageFields: image fields supported on HMI</p> <p>mediaClockFormats: mediaClockFormats supported on HMI</p> <p>graphicSupported: if graphics are supported or not on HMI</p> <p>imageCapabilities: image fields supported on hmi and their resolutions</p>	
audioPassThruCapabilities	samplingRate bitsPerSample audioType	Capabilities for audio capturing from the microphone (PerformAudioPassThru API)	
hmiZoneCapa		FRONT or	

bilities		REAR display type	
softButtonCap abilities	shortPress Available longPress Available upDownAv ailable imageSupported	Button actions supported on current HMI Define if images are supported for softButtons or not	

15.3.2.2 VR

To apply VR capabilities the data received in a response from [VR.GetCapabilities](#) is used. In case no answer from HMI or unavailability to get this data, the default VR capabilities will be applied on SDL.

Param Name	Description	Reference
capabilities	The VR capabilities supported by VR engine. See all possibly supported list in 9.2.3.2 VrCapabilities	To define all VR supported capabilities parameters see an example of hmi_capabilities.json section VR
Language	Current VR language	
languages	Languaged supported on VR engine	

15.3.2.3 TTS

To apply TTS capabilities the data received in a response from [TTS.GetCapabilities](#) is used. In case no answer from HMI or unavailability to get this data, the default TTS capabilities will be applied on SDL.

Param Name	Description	Reference
capabilities	Type of TTS phonems accepted by TTS engine	To define all TTS supported capabilities parameters see an hmi_capabilities.json section TTS
Language	Current TTS language	
languages	Languaged supported on TTS	

15.3.2.4 Buttons

To apply Buttons capabilities the data received in a response from [Buttons.GetCapabilities](#) is used. In case

no answer from HMI or unavailability to get this data, the default Buttons capabilities will be applied on SDL.

Param Name		Description	Reference
capabilities	name shortPressAvailable longPressAvailable upDownAvailable	An array of buttons names supported and appropriate pressTypes available	To define all Buttons supported capabilities parameters see an example of hmi_capabiliteis.json section Buttons
presetBankCapabilities	onScreenPresetsAvailable	Availability of on-screen presets on a current display	

15.3.2.5 VehicleInfo

To get Vehicle info data which is requested by RegisterApplInterface mobile API, [VehicleInfo.GetVehicleType](#) is sent from SDL to HMI. In case no answer from HMI or unavailability to get this data, the default Vehicle info data must be applied on SDL.

Param Name	Description	Reference
make	Make of the vehicle SDL is running on	See an example of hmi_capabiliteis.json section VehicleInfo
model	Model of the vehicle SDL is running on	
modelYear	modelYear of the vehicle SDL is running on	
trim	Trim of the vehicle	

15.3.2.6 SyncMessageVersion

SyncMessageVersion data is used for communication between SDL and mobile application to notify an application about supported protocol ranges on HeadUnit system. At the moment the parameter has no reference to HMI API and completely relates to Application<->SDL interaction.

Param Name	Description	Reference
majorVersion	Defines the major protocol	See an example of hmi_capabiliteis.json

	version supported by SDL	section SyncMessageVersion
minorVersion	Defines the minor protocol version supported by SDL	

15.3.3 Example

The body of the hmi_capabilities.json file:

```
{
  "UI":
  {
    "language": "EN_US",
    "languages": [
      "EN_US", "ES_MX", "FR_CA", "DE_DE", "ES_ES", "EN_GB",
      "RU_RU", "TR_TR", "PL_PL", "FR_FR", "IT_IT", "SV_SE", "PT_PT",
      "NL_NL", "ZH_TW",
      "JA_JP", "AR_SA", "KO_KR", "PT_BR", "CS_CZ", "DA_DK",
      "NO_NO"
    ],
    "displayCapabilities":
    {
      "displayType": "GEN2_8_DMA",
      "keyboardProperties": {
        "name": "KeyboardProperties",
        "language": "EN-US",
        "keyboardLayout": "QWERTY",
        "keypressMode": "SINGLE_KEYPRESS"
      },
      "textFields": [
        {
          "name": "mainField1",
          "characterSet": "TYPE2SET",
          "width": 500,
          "rows": 1
        },
        {
          "name": "mainField2",
          "characterSet": "TYPE2SET",
          "width": 500,
          "rows": 1
        },
        {
          "name": "mainField3",
          "characterSet": "TYPE2SET",
          "width": 500,
          "rows": 1
        },
        {
          "name": "mainField4",
          "characterSet": "TYPE2SET",
          "width": 500,
          "rows": 1
        }
      ]
    }
  }
}
```

```
        "name": "statusBar",
        "characterSet": "TYPE2SET",
        "width": 500,
        "rows": 1
    },
    {
        "name": "mediaClock",
        "characterSet": "TYPE2SET",
        "width": 500,
        "rows": 1
    },
    {
        "name": "mediaTrack",
        "characterSet": "TYPE2SET",
        "width": 500,
        "rows": 1
    },
    {
        "name": "alertText1",
        "characterSet": "TYPE2SET",
        "width": 500,
        "rows": 1
    },
    {
        "name": "alertText2",
        "characterSet": "TYPE2SET",
        "width": 500,
        "rows": 1
    },
    {
        "name": "alertText3",
        "characterSet": "TYPE2SET",
        "width": 500,
        "rows": 1
    },
    {
        "name": "scrollableMessageBody",
        "characterSet": "TYPE2SET",
        "width": 500,
        "rows": 1
    },
    {
        "name": "initialInteractionText",
        "characterSet": "TYPE2SET",
        "width": 500,
        "rows": 1
    },
    {
        "name": "navigationText1",
        "characterSet": "TYPE2SET",
        "width": 500,
        "rows": 1
    },
    {
        "name": "navigationText2",
        "characterSet": "TYPE2SET",
        "width": 500,
        "rows": 1
    }
},
```

```
        "name": "ETA",
        "characterSet": "TYPE2SET",
        "width": 500,
        "rows": 1
    },
    {
        "name": "totalDistance",
        "characterSet": "TYPE2SET",
        "width": 500,
        "rows": 1
    },
    {
        "name": "navigationText",
        "characterSet": "TYPE2SET",
        "width": 500,
        "rows": 1
    },
    {
        "name":
"audioPassThruDisplayText1",
        "characterSet": "TYPE2SET",
        "width": 500,
        "rows": 1
    },
    {
        "name":
"audioPassThruDisplayText2",
        "characterSet": "TYPE2SET",
        "width": 500,
        "rows": 1
    },
    {
        "name": "sliderHeader",
        "characterSet": "TYPE2SET",
        "width": 500,
        "rows": 1
    },
    {
        "name": "sliderFooter",
        "characterSet": "TYPE2SET",
        "width": 500,
        "rows": 1
    },
    {
        "name": "notificationText",
        "characterSet": "TYPE2SET",
        "width": 500,
        "rows": 1
    },
    {
        "name": "menuName",
        "characterSet": "TYPE2SET",
        "width": 500,
        "rows": 1
    },
    {
        "name": "secondaryText",
        "characterSet": "TYPE2SET",
        "width": 500,
        "rows": 1
    },
    {
        "name": "thirdText"
    }
},
```

```
{
    "name": "tertiaryText",
    "characterSet": "TYPE2SET",
    "width": 500,
    "rows": 1
},
{
    "name": "timeToDestination",
    "characterSet": "TYPE2SET",
    "width": 500,
    "rows": 1
},
{
    "name": "turnText",
    "characterSet": "TYPE2SET",
    "width": 500,
    "rows": 1
},
{
    "name": "menuTitle",
    "characterSet": "TYPE2SET",
    "width": 500,
    "rows": 1
}
],
"imageFields":
[
{
    "name": "softButtonImage",
    "imageTypeSupported": [
        [
        ],
        "imageResolution": {
            "resolutionWidth": 35,
            "resolutionHeight": 35
        }
    },
    {
        "name": "choiceImage",
        "imageTypeSupported": [
            [
            ],
            "imageResolution": {
                "resolutionWidth": 35,
                "resolutionHeight": 35
            }
        },
        {
            "name": "choiceSecondaryImage",
            "imageTypeSupported": [
                [
                ],
                "imageResolution": {
                    "resolutionWidth": 35,
                    "resolutionHeight": 35
                }
            }
        }
    }
]
```

```
        "resolutionWidth":35,  
  
        "resolutionHeight":35  
    },  
},  
{  
  
    "name":"menulcon",  
    "imageTypeSupported":  
    [  
    ],  
    "imageResolution":  
    {  
        "resolutionWidth":35,  
  
        "resolutionHeight":35  
    },  
},  
{  
  
    "name":"cmdlcon",  
    "imageTypeSupported":  
    [  
    ],  
    "imageResolution":  
    {  
        "resolutionWidth":35,  
  
        "resolutionHeight":35  
    },  
},  
{  
    {  
        "name":"applcon",  
        "imageTypeSupported":  
        [  
        ],  
        "imageResolution":  
        {  
            "resolutionWidth":35,  
            "resolutionHeight":35  
        }  
    },  
},  
{  
    "name":"graphic",  
    "imageTypeSupported":  
    [  
    ],  
    "imageResolution":  
    {  
        "resolutionWidth":35,  
        "resolutionHeight":35  
    }  
},  
{  
    "name":"locationlImage",  
    "imageTypeSupported":  
    [  
        "GRAPHIC_PNG"  
    ],  
    "imageResolution":
```

```
        {
          "resolutionWidth":35,
          "resolutionHeight":35
        }
      },
      "mediaClockFormats":
      [
        "CLOCK1","CLOCK2","CLOCK3","CLOCKTEXT1"
      , "CLOCKTEXT2","CLOCKTEXT3","CLOCKTEXT4"
      ],
      "graphicSupported":true,
      "templatesAvailable":
      [
        "DEFAULT","MEDIA","NON-MEDIA","ONSCREEN_PRESETS","NAV_FULLSCREEN_MAP","NAV_KEYBOARD",
        "GRAPHIC_WITH_TEXT","TEXT_WITH_GRAPHIC","TILES_ONLY","TEXTBUTTONS_ONLY",
        "GRAPHIC_WITH_TILES","TILES_WITH_GRAPHIC","GRAPHIC_WITH_TEXT_AND_SOFTBUTTONS",
        "TEXT_AND_SOFTBUTTONS_WITH_GRAPHIC","GRAPHIC_WITH_TEXTBUTTONS",
        "TEXTBUTTONS_WITH_GRAPHIC","LARGE_GRAPHIC_WITH_SOFTBUTTONS",
        "DOUBLE_GRAPHIC_WITH_SOFTBUTTONS","LARGE_GRAPHIC_ONLY"
      ],
      "screenParams":
      {
        "resolution":
        {
          "resolutionWidth":800,
          "resolutionHeight":350
        },
        "touchEventAvailable":
        {
          "pressAvailable":true,
          "multiTouchAvailable":false,
          "doublePressAvailable":false
        }
      },
      "numCustomPresetsAvailable":8,
      "imageCapabilities":
      [
        "DYNAMIC",
        "STATIC"
      ],
      "audioPassThruCapabilities":
      {
        "samplingRate" : "44KHZ",
```

```
        "bitsPerSample" : "RATE_8_BIT",
        "audioType" : "PCM"
    },
    "hmiZoneCapabilities": "FRONT",
    "softButtonCapabilities":
    [
        {
            "shortPressAvailable": true,
            "longPressAvailable": true,
            "upDownAvailable": true,
            "imageSupported": true
        }
    ]
},
"VR":
{
    "capabilities": ["TEXT"],
    "language": "EN_US",
    "languages":
    [
        "EN_US", "ES_MX", "FR_CA", "DE_DE", "ES_ES", "EN_GB",
        "RU_RU", "TR_TR", "PL_PL", "FR_FR", "IT_IT", "SV_SE", "PT_PT", "NL_NL", "ZH_TW",
        "JA_JP", "AR_SA", "KO_KR", "PT_BR", "CS_CZ", "DA_DK", "NO_NO"
    ]
},
"TTS":
{
    "capabilities": "TEXT",
    "language": "EN_US",
    "languages":
    [
        "EN_US", "ES_MX", "FR_CA", "DE_DE", "ES_ES", "EN_GB",
        "RU_RU", "TR_TR", "PL_PL", "FR_FR", "IT_IT", "SV_SE", "PT_PT", "NL_NL", "ZH_TW",
        "JA_JP", "AR_SA", "KO_KR", "PT_BR", "CS_CZ", "DA_DK", "NO_NO"
    ]
},
"Buttons":
{
    "capabilities":
    [
        {
            "name": "PRESET_0",
            "shortPressAvailable": true,
            "longPressAvailable": true,
            "upDownAvailable": true
        },
        {
            "name": "PRESET_1",
            "shortPressAvailable": true,
            "longPressAvailable": true,
            "upDownAvailable": true
        },
        {
            "name": "PRESET_2",
            "shortPressAvailable": true,
            "longPressAvailable": true,
            "upDownAvailable": true
        }
    ]
}
```

```
        "longPressAvailable" :true,
        "upDownAvailable"   :true
    },
    {
        "name":"PRESET_3",
        "shortPressAvailable":true,
        "longPressAvailable" :true,
        "upDownAvailable"   :true
    },
    {
        "name":"PRESET_4",
        "shortPressAvailable":true,
        "longPressAvailable" :true,
        "upDownAvailable"   :true
    },
    {
        "name":"PRESET_5",
        "shortPressAvailable":true,
        "longPressAvailable" :true,
        "upDownAvailable"   :true
    },
    {
        "name":"PRESET_6",
        "shortPressAvailable":true,
        "longPressAvailable" :true,
        "upDownAvailable"   :true
    },
    {
        "name":"PRESET_7",
        "shortPressAvailable":true,
        "longPressAvailable" :true,
        "upDownAvailable"   :true
    },
    {
        "name":"PRESET_8",
        "shortPressAvailable":true,
        "longPressAvailable" :true,
        "upDownAvailable"   :true
    },
    {
        "name":"PRESET_9",
        "shortPressAvailable":true,
        "longPressAvailable" :true,
        "upDownAvailable"   :true
    },
    {
        "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
},
"VehicleInfo":
{
    "make"      :"Ford",
    "model"     :"Fiesta",
    "modelYear" :"2013",
    "trim"      :"SE"
},
"SyncMessageVersion":
{
    "majorVersion": 3,
    "minorVersion": 0
}
}
```

References

- 1) JSON-RPC 2.0 Specification:
<http://www.jsonrpc.org/specification>

Change History

Version	Date	Status	Change description	Author/Editor
V.1.0	07/27/2015	Initial Version	Draft for GENIVI	E.Saenko
CW31	07/31/2015		1) Section #7.18 PerformAudioPassThru - update description with audio capturing information - added diagram 2) Section 13.1 SDL.ActivateApp - update according to Genivi requirements - removed invalid diagrams 3) Section #15.1.2.2 MAIN - added AppStorageFolder example 4) Section #15.1.2.1 HMI - update description - removed extra parameters 5) Section #15.1.2.3 LOGGING - removed as not related to GENIVI 6) Section 15.2.2.4 MEDIA MANAGER - removed VideoStreamConsumer, AudioStreamConsumer, MQAudioPath 7) Section #15.2 HMI<->SDL fake parameters processing added	E.Saenko
CW32	2015/08/07	Draft	1) Section #7.12.2.2 SetGlobalProperties Parameters - added reference for keyboardProperties to hmi_capabilities.json section 2) Section #8.1 Buttons.GetCapabilities - added SDL note about hmi_capabilities.json 3) Section #9.2 VR.GetCapabilities - updated SDL note about hmi_capabilities.json 4) Section #10.2 TTS.GetCapabilities - updated SDL note about hmi_capabilities.json 5) Section #15.1.2.1 smartDeviceLink.ini HMI - make reference to appropriate chapters in Desctiption - added LinkToWebHMI parameter 6) Section #15.1.2.2 smartDeviceLink.ini MAIN	E.Saenko

			<ul style="list-style-type: none">- added SDLVersion parameter- added AppResourceFolder parameter- updated AppStorageFolder parameter description <p>Section #15.1.2.4 smartDeviceLink.ini MEDIA MANAGER</p> <ul style="list-style-type: none">- updated description for NamedVideoPipePath and NamedAudioPipePath <p>7) Section #15.3 HMI Capabilities JSON file added</p>	
--	--	--	---	--