



Baidu CarLife Integration Specification



Content

BAIDU CARLIFE INTEGRATION SPECIFICATION.....	1
CONTENT.....	2
1. VERSION HISTORY.....	6
2. FOREWORD.....	10
2.1. BACKGROUND.....	10
2.2. GENERAL FRAMEWORK OF CARLIFE ON HEAD UNIT.....	11
2.2.1. Transmission Medium.....	11
2.2.2. Overall Framework of CarLife.....	11
2.2.3. Structure of CarLife on HU.....	11
2.3. DATA TRANSMISSION MEDIUM-PB.....	12
2.4. DEFINITIONS.....	13
3.HARDWARE REQUIREMENTS.....	14
3.1. THE MINIMUM HARDWARE REQUIREMENTS OF HEAD UNIT.....	14
3.2. REQUIREMENTS OF MD FOR CARLIFE.....	15
4. HU CAPABILITY CLOSELY RELATED TO CARLIFE.....	15
5. DATA CHANNELS.....	16
6. CONNECTION ENGINE.....	17
6.1. CONNECTION METHOD.....	17
6.1.1. AOA Connection.....	17
6.1.2. EAP Connection.....	29
6.2. BLUETOOTH AUTO-PAIR (HU CAN CHOOSE WHETHER TO SUPPORT THIS FUNCTION).....	51
6.2.1. BT Auto-pair Mode.....	51
6.2.2. Auto-pair Sequence.....	52
6.2.3. Auto-pair Initialization.....	54
6.2.4. Auto-pair Interface Definition.....	56
7. VIDEO ENGINE.....	58
7.1. PROCESSING OF BDSC.....	58
7.2. DYNAMIC FPS MECHANISM.....	59
7.2.1. FPS Adjustment on HU.....	59
7.2.2. FPS Adjustment on MD.....	59
7.3. DYNAMIC RESOLUTION MECHANISM.....	59
7.4. ADAPTATION STRATEGY FOR EXTRA-WIDE OR EXTRA-HIGH SCREENS.....	60
8. AUDIO ENGINE.....	61
8.1. CARLIFE AUDIO ENGINE ARCHITECTURE.....	61
8.2. AUDIO STREAM TYPES.....	62



8.3.	AUDIO FOCUS TYPES.....	64
8.4.	AUDIO FOCUS MANAGEMENT STRATEGY.....	65
8.4.1.	Duck Process.....	66
8.4.2.	Mute Strategy.....	67
8.5.	STRATEGY IN CONFLICT OF HU NATIVE AUDIO AND CARLIFE AUDIO.....	67
8.6.	AUDIO SERVICES PRIORITY INSTRUCTION.....	68
8.7.	VOICE RECOGNITION.....	68
9.	BLUETOOTH TELEPHONE.....	70
9.1.	FOREWORD.....	70
9.2.	INTERFACE STRUCTURE.....	71
9.2.1.	Overall Structure.....	71
9.2.2.	Bluetooth Telephone Message Sequence.....	72
9.3.	INTERFACE DEFINITION.....	74
9.3.1	Telephone Functions.....	74
9.3.2.	Detailed Interface Definitions.....	74
9.4.	INTERACTION SEQUENCE OF CRITICAL SCENARIOS.....	79
9.4.1.	Outgoing Call scenario.....	79
9.4.2.	Sending DTMF Code During Phone call.....	80
9.4.3.	Scenario When Bluetooth Status Changes.....	80
10.	COMMUNICATION PROTOCOL.....	81
10.1.	NAMING RULES OF SERVICE TYPE.....	81
10.2.	COMMAND MESSAGE.....	82
10.2.1.	Package Structure.....	82
10.2.2.	Related Messages.....	83
10.2.3.	Sequence Diagram.....	109
10.3.	VIDEO MESSAGE.....	110
10.3.1.	Package Structure.....	111
10.3.2.	Related Messages.....	112
10.3.3	Sequence Diagram.....	115
10.4.	AUDIO MESSAGE.....	118
10.4.1.	Package Structure.....	118
10.4.2.	Related Messages.....	119
10.4.3.	Sequence Diagram.....	121
10.5.	VOICE MESSAGE.....	123
10.5.1.	Package Structure.....	123
10.5.2.	Related Messages.....	124
10.5.3.	Sequence Diagram.....	135
10.6.	BACK CONTROL MESSAGES.....	138
10.6.1.	Back Control Event Strategy.....	138
10.6.3.	Package Structure.....	144
10.6.4.	Related Messages.....	145
10.6.5.	Sequence Diagram.....	146



11. CARLIFE ON MD SPECIAL SCENE STRATEGY.....	147
11.1. SCREEN LOCK AND BACKGROUND/FOREGROUND SWITCH IN CARLIFE APP.....	148
11.2. MD PHONE CALL PROCESS.....	148
12. USE OF HU DATA AND SUGGESTIONS.....	149
12.1. MSG_CMD_CAR_VELOCITY.....	154
12.2. MSG_CMD_CAR_GPS.....	154
12.3. MSG_CMD_CAR_GYROSCOPE.....	154
12.4. MSG_CMD_CAR_ACCELERATION.....	155
12.5. MSG_CMD_CAR_OIL.....	156
12.6. MSG_CMD_CAR_DATA_GEAR.....	156
12.7. MSG_CMD_CAR_DATA_SUBSCRIBE.....	156
12.8. MSG_CMD_CAR_DATA_SUBSCRIBE_DONE.....	156
12.9. MSG_CMD_CAR_DATA_SUBSCRIBE_START.....	158
12.10. MSG_CMD_CAR_DATA_SUBSCRIBE_STOP.....	159
13. DATA PROVIDED TO HU FROM CARLIFE.....	160
12.1. NAVIGATION TBT INFO.....	160
12.2. MUSIC PLAY INFO.....	166
12.3. MUSIC PLAY PROGRESS INFO.....	167
14. EXITING MECHANISM OF CARLIFE ON HU.....	167
15. UPDATE STRATEGY.....	167
15.1. CARLIFE ON HU.....	167
15.2. CARLIFE ON MD.....	168
15.3. COMPATIBILITY STRATEGY.....	168
15.3.1. CarLife on HU.....	168
15.3.2. CarLife on MD.....	168
15.3.3. Data Protocol.....	168
16. SAFETY VERIFICATION.....	169
16.1. SAFETY VERIFICATION OF CARLIFE ON MD TO CARLIFE ON HU.....	169
16.1.1. Definition of Verification Message.....	169
16.1.2 Verification Process.....	169
17. PERFORMANCE INDICATORS.....	170
18. STATICSTIC INFO CLARIFICATION.....	171
18.1. CARLIFE ON HU STATISTIC INFO CLARIFICATION (MSG_CMD_HU_INFO).....	171
18.2. CHANNEL AND RUNNING STATISTIC INFO CLARIFICATION (MSG_CMD_STATISTIC_INFO)	
.....	172
19. SUPPORT FROM CARLIFE TEAM TO HU OEMS.....	173
19.1 KEY TEST TOOLS.....	173
19.1.1 CarLife Protocol Tools.....	173



20. CARLIFE CERTIFICATION.....	174
20.1. PARTNERS SELF-TEST AND LAUNCH ACCEPTANCE PROCESS.....	174
20.2. CERTIFICATION ACCEPTANCE BY BAIDU.....	176
21. HU HMI DESIGN STANDARDS.....	177
21.1. THE STANDARD FOR CARLIFE ICON AND APP TITLE ON HU.....	178
21.2. THE STANDARD OF CONNECTION RELATED INTERACTION ON HEAD UNIT.....	179
21.2.1. Before Connection.....	179
21.2.2. No Installation of CarLife on MD.....	180
21.2.3. Connecting.....	180
21.2.4. Connected Successfully.....	181
21.2.5. Connection Failed/Disconnected.....	182
21.2.6. Connection Help.....	182
21.2.7. CarLife in Background/Screen Locked.....	183
21.2.8. Exit/Return CarLife.....	184
21.2.9. Bluetooth NOT Connected.....	184
21.3. PHONE RELATED INTERACTION.....	185
21.3.1. Bluetooth Connected.....	185
21.4. THE STANDARD OF PARTNERS' BRAND CUSTOMIZATION ON HU.....	186
21.4.1. CarLife Launch Page on HU.....	186
21.4.2. OEMs Button.....	187
21.5. ACTIVATING CARLIFE BY DEFAULT.....	187

1. Version History

Update Record

Version	Description	Date	Modifier
V1.0	Creation	2015/12/09	Carlie Team
V.1.0.1	-Update Bluetooth Auto-pair and Bluetooth phone chapters, such as add Auto-pair Initialization procedure and one new command channel message	2015/12/17	Yang Jie
V.1.0.2	-Add BT address filed in messages: MSG_CMD_BT_HFP_INDICATION MSG_CMD_HU_INFO MSG_CMD_MD_INFO	2016/1/5	Yang Jie
V.1.0.3	-Add three BT related command messages: MSG_CMD_BT_HFP_RESPONSE, MSG_CMD_BT_HFP_STATUS_REQUEST & MSG_CMD_BT_HFP_STATUS_RESPONSE	2016/1/5	Yang Jie
V.1.0.4	-Add HU functionality customized messages and update connection Sequence Diagram: MSG_CMD_MD_FEATURE_CONFIG_REQUEST MSG_CMD_HU_FEATURE_CONFIG_RESPONSE	2016/1/8	Ouyang Nengjun
V1.0.5	-Update the Auto-pair initialization Sequence Diagram in chapter 6.2.3, using MSG_CMD_BT_HFP_CONNECTION to inform HU the completion of identification. Update related definition of specific messages. -Add KEY CODE_NUMBER_PLUS -Add multi-call status of HU in message MSG_CMD_BT_HFP_INDICATION	2016/1/26	Yang Jie
V1.0.6	-Update Chapter 6.2.3 and the Sequence Diagram, add two messages MSG_CMD_BT_START_IDENTIFY_REQ, MSG_CMD_BT_IDENTIFY_RESULT_IND to the Command List of Chapter 9.3.2 and Chapter 10.2.2. Add BT name field on	2016/1/29	Yang Jie Zhang Ling



	MD_INFO and HU_INFO -Amend the description for CarLife Launch page in HMI		
V1.0.7	Update BT Auto-pair Sequence Diagram Figure 6.2-3, update description of MSG_CMD_BT_START_AUTOPAIR	2016/2/1	Yang Jie
V1.0.8	Adding the name and type of Bonjour service	2016/2/2	FangSheng
V1.0.9	Adding Feature_Config request and response in the Sequence Diagram of BT initialization. Delete MULTICALL_INACTIVE and add UNMUTE_MIC command	2016/2/25	Yang Jie
V1.1.0	Modify the service_type value of those command messages: MSG_CMD_BT_HFP_STATUS_RESPONSE\ MSG_CMD_MD_FEATRE_CONFIG_REQUE STMSG_CMD_HU_FEATURE_CONFIG_RES PONSEMSG_CMD_HU_AUTHEN_RESULT\ MSG_CMD_MD_AUTHEN_RESULT	2016/2/29	Zhao Ke
V1.1.1	Add a command message error code statistic message: MSG_CMD_ERROR_CODE	2016/2/29	Fang Sheng
V1.1.2	Modify table10.2-2,adding AUDIO_TRANSMISSION_MODE	2016/3/4	Liu Caiquan
V1.1.3	Add video Related Messages: MSG_CMD_VIDEO_ENCODER_JPEG, MSG_CMD_VIDEO_ENCODER_JPEG_ACK	2016/3/14	Song Ye
V1.1.4	Add BT Phone Related Messages: MSG_CMD_BT_HFP_CALL_STATUS_COVER	2016/3/16	Yang Jie
V1.1.5	Complete those messages: MSG_CMD_START_BT_AUTOPAIR, MSG_CMD_HU_BT_OOB_INFO, MSG_CMD_BT_IDENTIFY_RESULT_IND	2016/3/23	Yang Jie
V1.1.6	Add MSG_CMD_MD_EXIT	2016/3/23	Zhao Ke
V1.1.7	Modify the message transmitted via transparent method, defining as MSG_TOUCH_ACTION message	2016/3/24	Zhao Ke
V1.1.8	Modify MSG_CMD_BT_HFP_CONNECTION in 9.3.2.3 wherever is inconsistent with implantation code	2016/3/30	Yang Jie
V1.2	Add EAP in 6.1.3 and switch NVM of 6.1.4 to EAP	2016/5/24	Fangsheng
V.1.2.1	Modify recording quality requirements, Change amplitude to 5000.	2016/6/28	Ouyangnengjun
V1.2.2	Add heartbeat message from HU to MD in EAP	2016/7/7	Luo Xubin



	solution in section 6.1.3.7		
V1.2.3	Change “CarLife” to “Carlife”, “Message” to “message”, and modify the name of status in NaviStatus in Command Message. Correct coding error in chapter chart of Command Message	2016/07/29	Zhao Ke
V1.2.4	Add the description of MSG_VR_MODULE_STATUS	2016/08/30	Ouyang NengJun
V1.2.5	Delete MSG_CMD_TELE_STATE_CHANGE_INCOMING, MSG_CMD_TELE_STATE_CHANGE_OUTGOING, MSG_CMD_TELE_STATE_CHANGE_IDLE, MSG_CMD_TELE_STATE_CHANGE_INCALLING, which is replaced by MSG_CMD_MODULE_STATUS Delete CarLife on HU verifies CarLife on MD in Chapter 17	2016/10/31	Yang Jie Liu Caiquan
V1.2.6	Update the sections of BT phone call and BT auto-pair according to the latest implementation	2016/11/04	Yang Jie
V1.2.7	Update data subscribe, Add car GPS data supporting	2016/11/10	Yang Jie
V1.2.8	Modify CarLife Acceptance process fully functional acceptance part	2016/11/21	Long Huijuan
V1.2.9	Add Chapter 6.1.2.10, introducing the launch strategy for CarPlay and CarLife on MD	2016/12/20	Zhang Ling
V1.3.0	Statistic message connected to newly added HU MSG_CMD_HU_CONNECT_STATISTIC	2016/12/28	Zhou Xingjie
V1.3.1	Delete the NCM connection; for IOS, it is recommended to use EAP connection, and for Android, it is recommended to use AOA connection.	2017/1/23	Fang Sheng
V1.3.2	Add reserved field 0x0001005A~0x0001006F; change the ID of MSG_CMD_HU_CONNECT_STATISTIC to 0x00018070; and add HU type configuration item ENGINE TYPE	2017/3/6	Liu Caiquan
V1.3.3	Modify car-mounted GPS message (MSG_CMD_CAR_GPS) to remove unwanted fields	2017/3/10	Luo Xubin
V1.3.4	Add input method disabling configuration item according to vehicle speed (INPUT_DISABLE)	2017/4/11	Luo Xubin



V1.3.5	Modify car-mounted GPS message (MSG_CMD_CAR_GPS) and change the speed unit to m/s	2017/4/18	Luo Xubin
V1.3.6	Add audio interrupt message (MSG_VR_AUDIO_INTERRUPT) for audio channel	2017/4/20	Zhou Xingjie
V1.3.7	MSG_VR_MODULE_STATUS ID conflict modification MSG_CMD_MD_INFO command annotation	2017/6/2	Ouyang Nengjun
V1.3.8	Update the direction of the following messages: MSG_CMD_CAR_DATA_SUBSCRIBE & MSG_CMD_CAR_DATA_SUBSCRIBE_STOP. Update data files of the following 4 messages: MSG_CMD_CARLIFE_DATA_SUBSCRIBE, MSG_CMD_CARLIFE_DATA_SUBSCRIBE_DONE, MSG_CMD_CARLIFE_DATA_SUBSCRIBE_START, MSG_CMD_CARLIFE_DATA_SUBSCRIBE_STOP, to the same pb structure – CarlifeSubscribeMobileCarLifeInfoList & CarlifeSubscribeMobileCarLifeInfo	2017/6/20	Song Ye
V1.3.9	Modify the definition of enum NaviState (at two places)	2017/6/21	Liao Ruihua
V2.0.0	Add enum PBVersion definition to MSG_CMD_HU_PROTOCOL_VERSION, and the latest version of PBVersion is 2.0 Modify enum ModuleId definition and remarks of MSG_CMD_MODULE_STATUS Modify enum ModuleId definition and remarks of MSG_CMD_MODULE_CONTROL Add enum EDogState and enum CruiseState definition	2017/7/19	Liao Ruihua
V2.0.1	Modify the description of Internal UI outgoing call scenario in chapter 9.4.1. The original document description only applies to IOS.	2017/7/27	Zhou Xingjie
V2.1.0	1. Supplement Chapter 6.1.2.4, Alert is recommended for the launch of iOS App; 2. Add the description of how EAP connected HU data is read in Chapter 6.1.2.8; 3. Update the description of sequence of BT	2017/9/28	CarLife Team



	Auto-pair in Chapter 6.2; 4. Supplement the adaptation strategy for ultra-wide or ultra-high screens in Chapter 7.4; 5. Update the description of playing sequence of VR & VR-TTS in Chapter 10.5; 6. Update the description of HU acceptance procedure in Chapter 21.		
V2.1.3	Update 6.2, android 9.0 Explanation for not supporting Bluetooth auto-pairing.	2018/11/29	Wang Qing
V2.1.4	Add 6.1.1.7.3, method of judging low Android version. Update 7.5, add the countermeasure if the first I frame which HU received is blank. Update 9.1, clarify that BT address will not be supported by MD. Update 10.5.2, notes on VR Recognition.	2019/7/04	Zhou Xingjie
V2.1.5	Update 7.4, modify the adaptation strategy of the wide screen HU, and clarify how to realize wide screen adaptation.	2019/7/18	Luo Xubin

2. Foreword

2.1. Background

Baidu CarLife Integration Specification provides guidance to partners on aspects of hardware, system and software design in CarLife integration project.

CarLife is the Smartphone-Integration solution. Drivers of CarLife can share the applications suitable for safe driving conditions on MD (Mobile Device) to HU (Head Unit), through the multi-screen sharing and interaction technology, and use the Touch Screen, Hard Keys, Knob Control and Microphone to control CarLife. This solution takes full respective advantages of MD and HD, and provides safe and abundant infotainment experience to drivers.

All HMI in CarLife are strictly optimized according to vehicle regulations, suitable for driving conditions.

More information about CarLife are provided on official website: <http://carlife.baidu.com>



2.2. General Framework of CarLife on Head Unit

2.2.1. Transmission Medium

In CarLife, Mobile Device and Head Unit use the following physical transmission medium:

- USB cable (for Android smartphone) or iOS Lightning (for iPhone): CarLife uses USB port to transmit data interaction other than phone call function. It builds up TCP/IP layer based on this physical transmission medium.
- Bluetooth: Use HFP to pick up and hang up phones as well as other functions.
- **WiFi direct connection: Android smartphones support WiFi direct connection with HU to achieve data transmission other than phone call function.**

2.2.2. Overall Framework of CarLife

Figure 2.2-1 displays the overall framework of CarLife:

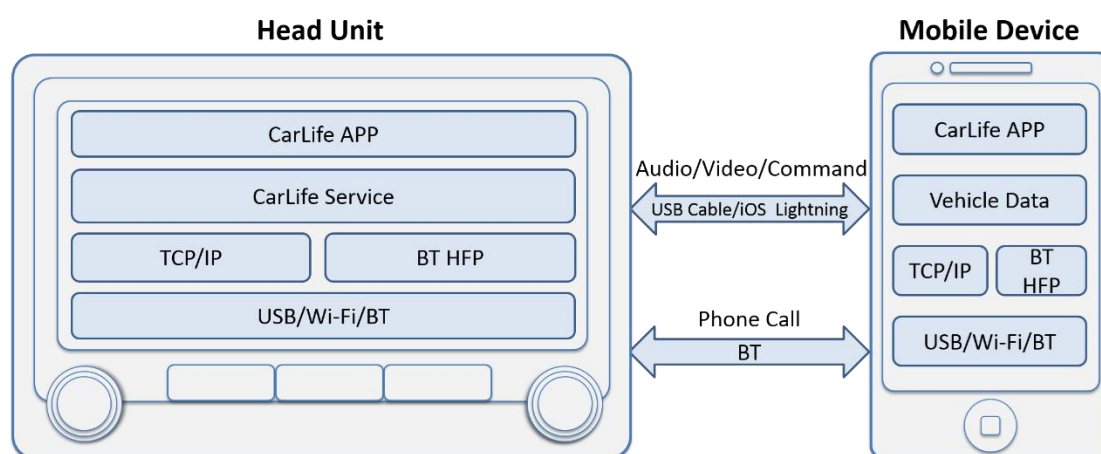


Figure 2.2-1: Picture of CarLife overall framework

2.2.3. Structure of CarLife on HU

CarLife on HU consists of HMI layer, the protocol layer and the related support hardware, which is shown in Figure 2.2-2:

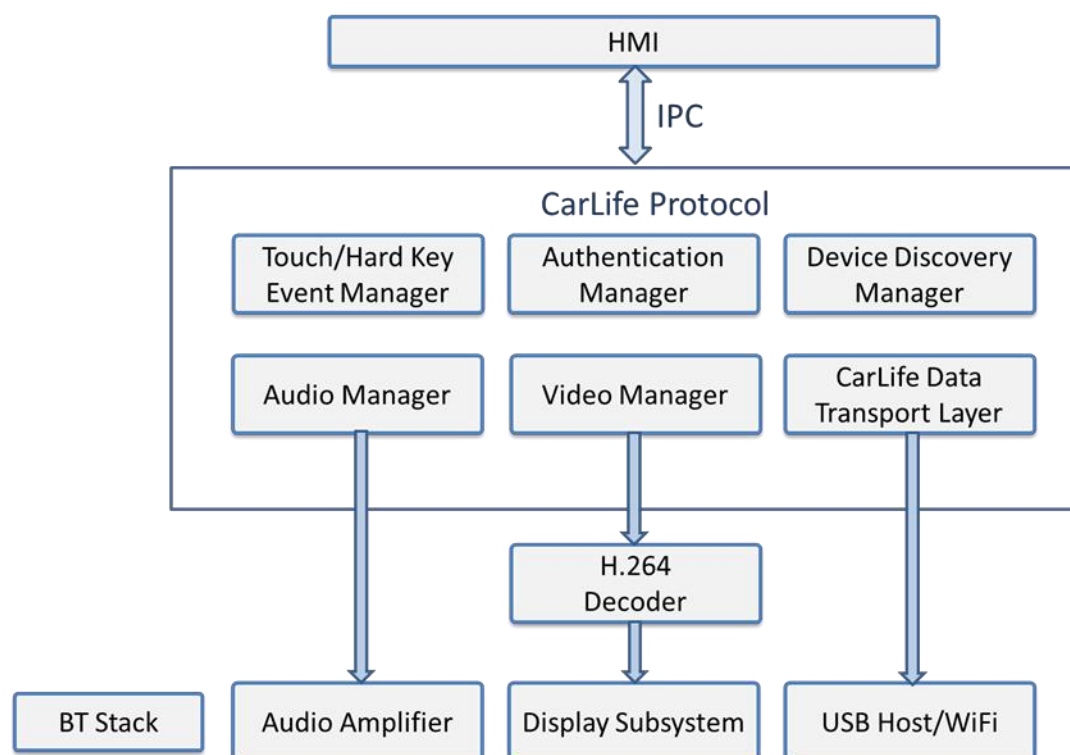


Figure 2.2-2: The Overall Framework of CarLife on Head Unit

The HMI layer presents CarLife contents generated from MD to drivers, eventually bringing more convenient and richer driving experience to users.

CarLife protocol layer mainly includes functions such as information obtainment from touch screen and physical buttons, device identification and authentication management between MD and HU of CarLife, as well as data transmission, video decoding and audio player management.

For hardware layer, CarLife on Head Unit requires hardware support from USB Host/Wi-Fi module, Bluetooth transmission module, video decode module and audio play module to implement respective functions of CarLife on HU according to CarLife protocol.

2.3. Data Transmission Medium-PB

Protocol buffer (PB in short) is a type of data exchangeable format, which is independent from the language program and platform. More importantly, PB makes sure that protocol of CarLife can be compatible.

As for the data protocol, CarLife implements three kinds of official languages (java, c++ and Objective-C) used in different scenes of Mobile Device and Head Unit. As a binary format, it is better and faster for exchanging data than xml. It can be used in distributed applications' data transmission or data exchange in heterogeneous environment. As a high efficient binary data transmission format with good compatibility, it is widely used in network transmission, file



configuration, data storage and so on.

The protobuf version of CarLife is 2.2.0.

2.4. Definitions

Name	Description
Head Unit	Vehicle entertainment infotainment system (Head Unit, HU in short).
CarLife on Head Unit	The CarLife app which runs on Head Unit.
Mobile Device	Mobile Device (MD in short) normally refers to Android smartphone and iPhone.
USB Cable	The term USB stands for "Universal Serial Bus". A USB port is a standard cable connection interface on personal computers and consumer electronics. USB ports allow stand-alone electronic devices to be connected via cables to a computer (or to each other).
iOS Lightning	iOS Lightning is a high-speed iPhone multi-function I/O port, namely, the official data wire for iPhone.
Audio Focus	Audio focus refers to the ownership audio modules (such as music, TTS prompt) obtain over speaker. A specific audio module has to apply for audio focus before using the speaker; if the application is successful, which means that the ownership of speaker is taken by the module, it can play; if the application is unsuccessful, the module cannot play. The ownership can be released according to requirement after successful play of the module.
Partners	OEMs and Suppliers
Quantized Amplitude	Quantized amplitude refers to the digital quantized value after modular conversion,



	namely, the quantized value of analog signal after its conversion to digital signal. For example: if the digital quantization accuracy is 16bit, for digital signal, the max voltage of audio analog signal is 5V (reference voltage 5V), then the digital signal value after modular conversion is 32768; if the min voltage of audio analog signal is 0V, then the digital signal value after modular conversion is 0.
--	--

Table 2.4-1 Definitions

3. Hardware Requirements

In order to ensure best user experience of CarLife, there are some corresponding requirements in software and hardware with Mobile Device and Head Unit which run CarLife.

Please aware the minimum hardware requirements of Head Unit cover the vast majority of current equipment on the market; generally OEMs or suppliers do not need to make any customization.

3.1. The Minimum Hardware Requirements of Head Unit

Item	Requirement
Physical Connection	USB 2.0 HiSpeed Host: • Standard USB A port • 480 Mbits/s throughput • 4.75~5.25V @ 500mA power to MD Bluetooth: • HandsFree Profile 1.5 • Secure Simple Pairing
High Resolution Display	• Minimum resolution: 800x480 pixels(recommend) • Close to 1:1 pixel aspect ratio (square pixels)
Video	H.264 hardware Decoding: • H.264 Base Line Profile level 3.1 is Required (Both sides to guarantee 800x480, 60fps; For 720p 30fps)
Audio	Supported Formats: PCM



	Audio data Streams: • Voice stream(Mobile Device to HU)&&Navigation TTS • Media stream • Bluetooth in-call audio • VR (HU to Mobile Device)
User Input Devices	Below physical or soft buttons are supported: • Touch screen • Knobs, buttons, • Physical or soft “Home” button to switch Car APP and Native APPs
Speakers	The car must provide: • Audio output via speakers
MFI	MFI certification chip USB OTG switch • Usb host mode • Usb device mode
Car Sensors (optional)	Below car data is required: • Car speed • Gro • Accelerometer • GPS
MIC (optional)	Below are Required for VR function: • Voice PCM flow • 16bit • amplitude: about 5000 • SNR > 12dB

Table 3.1-1 The Minimum Hardware Requirements of Head Unit

3.2. Requirements of MD for CarLife

- Android smartphone: Android 4.1 and beyond
- iPhone: iPhone5 and beyond; iOS 8.0 and beyond

4. HU Capability Closely Related to CarLife

- H.264 hardware decoder

When CarLife is running, Head Unit will use the H.264 hardware decoder to receive and decode video.



- Microphone

When using the voice recognition function, CarLife will use the microphone from HU to record. However, if Head Unit cannot support this function, it needs to notify the Mobile Device through the relevant protocol. About the relevant protocol, please read "12.5 Voice Message" .

- Speakers

CarLife will use the Speaker on Head Unit when applying the Audio Player and phone call function.

- Native Phone Call UI

If HUconnected MD via Bluetooth, when using the phone call function, it will ask HU to block native phone call HMI. As a result, all the phone call UI of Head Unit will be managed by CarLife. When drivers do not start CarLife, the native phone call UI will not be influenced.

- Audio Focus

In the CarLife audio engine, it will apply and release the audio focus of Head Unit according to different scenes.

5. Data Channels

CarLife protocol specifies five types of data streams including command stream (Cmd), video stream (Video), audio stream (Audio), speech stream (VR) and back control stream (Touch). Moreover, audio stream includes Media and Navigation TTS. This protocol defines six TCP socket channels to transmit these data streams:

- Command stream (Cmd): One tcp socket channel which is mainly used to transmit related command messages between Head Unit and Mobile Device, such as vehicle data, Bluetooth automatic pairing messages and video initialization message and so on.
- Video stream (Video): One tcp socket channel which is mainly used to transmit encoded video data to Head Unit.
- Audio stream (Audio): Two tcp socket channels, one is used to transmit Media audio stream, while the other to transmit navigation TTS (Text to Sound) audio stream. These two channels are used to transmit music and navigation voice broadcast in PCM stream respectively.
- Voice stream (VR): One tcp socket channel which is used to transmit users' voice data



from mic record on Head Unit to Mobile Device.

- Back control stream (Touch): One tcp socket channel which is used to transmit touch control or press control to Mobile Device.

There are different ways to set up these six channels according to different connection modes.

Details will be described in the connection engine section.

6. Connection Engine

6.1. Connection Method

6.1.1. AOA Connection

6.1.1.1. A Brief Introduction to AOA

For Android 5.0 and beyond, connection can be set up via AOA (Android Open Accessory). AOA is a kind of accessory communication mechanism designed by Google for Android system. This mechanism enables a communication with accessory through Bluetooth or USB, and Android system does not need to get any authority (like root authority or ADB debug authority). As for CarLife, MD does not need to get root right or turn on ADB debug mode, but accessory should possess the right of operating USB port. That is, in CarLife, the HU needs to possess the right of operating USB port. AOA connection can be supported on system of Android 3.1 (API level 12) or beyond, and it can also be supported as an extra lib on Android 2.3.4(API level 10). AOA 2.0 can be directly supported on system of Android 4.1 (API level 16) or beyond.

On Android Mobile Device, Android system provides the standardized Accessory API to access accessory. In USB communication, accessory USB acts as Host, while Android Mobile Device acts as Device.

Google officially defines Android Open Accessory Protocol as below:

<https://source.android.com/accessories/protocol.html>

Google officially provides adk2012 development kits (which includes the source code on Android side and Accessory side, please refer to Arduino platform)

<http://developer.android.com/tools/adk/index.html>



At present, MD Android5.0 or beyond accounts for a higher proportion, so it is recommended to default AOA connection or give priority to AOA connection; if the connection is successful, check the MD system's version number first, and where the MD system's version is below Android5.0, switch to ADB mode automatically.

6.1.1.1.1. Android USB Communication Mode

Android system supports multiple USB accessories. In terms of the role that Android Mobile Device plays in USB communication, there are Host Mode and Accessory Mode in Android USB communication. Details can be seen at Google official website:

<http://developer.android.com/guide/topics/connectivity/usb/index.html>

➤ Host Mode

Host mode refers that Android Mobile Device acts as USB host to power up the bus. Within this mode, Android Mobile Device needs to support USB Host and OTG functions. At this time, the USB Host of Android Mobile Device is called USB EH (Embedded Host). Compared with USB Host on PC, EH may be unable to load the drive program for unidentified accessory connected with the bus, thus, they define the Target Peripheral List. Those accessories mainly includes HID (Human Interface Device), BOMS (Bulk Only Mass Storage, for example, U-disk) and CDC (Communication Device Class, for instance, printer). The drive program of these accessories has already existed in the Android system (Linux Kernel), thus, Android Mobile Device could directly conduct data communication with them.

➤ Accessory Mode

Accessory mode refers that Android Mobile Device acts as USB Device, and accessory (like HU) acts as Host to power up the bus. Within this mode, the USB accessory is called Android accessory. This mode enables those Android Mobile Device (without Host mode) to interact with USB Accessory. Both Android Mobile Device and Android accessory should support AOA protocol. As for accessories (like HU) that do **NOT** support AOA protocol, it can connect with Android Mobile Device through Android ADK, thus acting as indirect accessory of Android Mobile Device.

The schematic diagram of Host mode and Accessory mode is displayed as below:

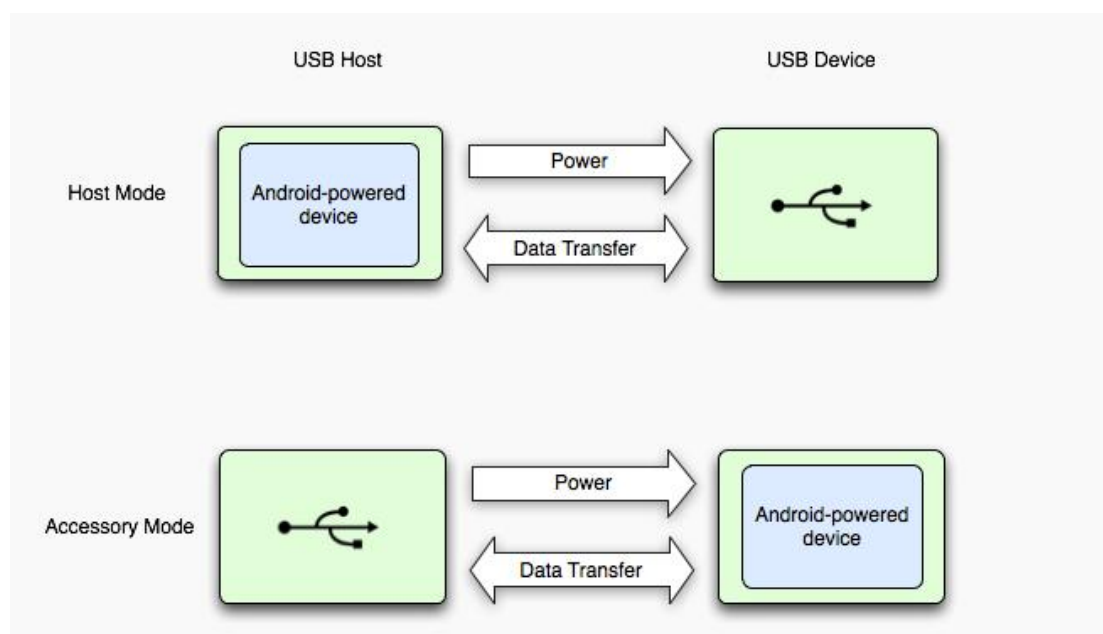


Figure 6.1-1: Host Mode and Accessory Mode

Host mode and Accessory mode are directly supported on platform of Android 3.1(API level 12) or beyond. Accessory mode could be supported on platform of Android 2.3.4 (API level 10) as an extra lib. The manufacturers (of HU) have choices in whether incorporating extra lib in Linux system mirror. Besides, whether supporting Host or Accessory mode depends on the hardwares of the **accessory (like HU)** but do NOT depend on the system version of Android Mobile Device that decide.

6.1.1.1.2. Android Accessory, ADK and AOA protocol

Android accessory refers the hardware that conforms to AOA protocol. It is a concept in Android accessory mode. In terms of hardware requirements, Android accessory should possess the microprocessor with USB EH function and conforming to AOA protocol. Also, it should possess input/output interfaces (including USB interface). Android accessory should work together within Android system.

ADK (Accessory Development Kit) is a development kit for reference in developing Android accessory, which is based on Arduino Open-source Electronics Prototyping Platform. ADK covers Hardware Design Documents of Android accessory, Source Code of Android Accessory Firmware and Source Code of Android Mobile Device Program.

AOA protocol is a connection protocol that needs to be obeyed when connecting Android Mobile Device with Android accessory via USB. This protocol specifies how the Android



accessory identifies Android Mobile Device and makes communication with it. AOA protocol also defines that the accessory should possess the following four functions:

1. Detect the USB interface of the accessory itself, and wait for connecting the Android Mobile Device;
2. Check if the Android Mobile Device support AOA protocol;
3. If necessary, launch USB Accessory mode on Android Mobile Device;
4. Establish USB connection with Android Mobile Device.

6.1.1.2. AOA Device Handshaking

6.1.1.2.1. AOA Device Handshaking Process

AOA protocol defines the handshaking process of Android Mobile Device and accessory:

1. Connect Android Mobile Device with accessory through USB

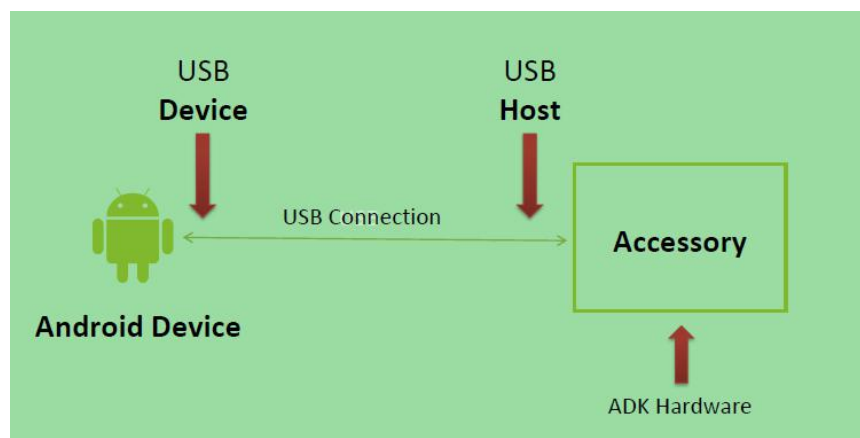


Figure 6.1-2: AOA Device Handshaking Process (1)

2. Accessory enumerates the Mobile Device that has been connected, judging if the Android Mobile Device is currently in Accessory Mode according to VID and PID. If not, the accessory will send out the request for switching to AOA mode, then it will continue to follow step 3; if it is in Accessory Mode, it will directly follow step 5.

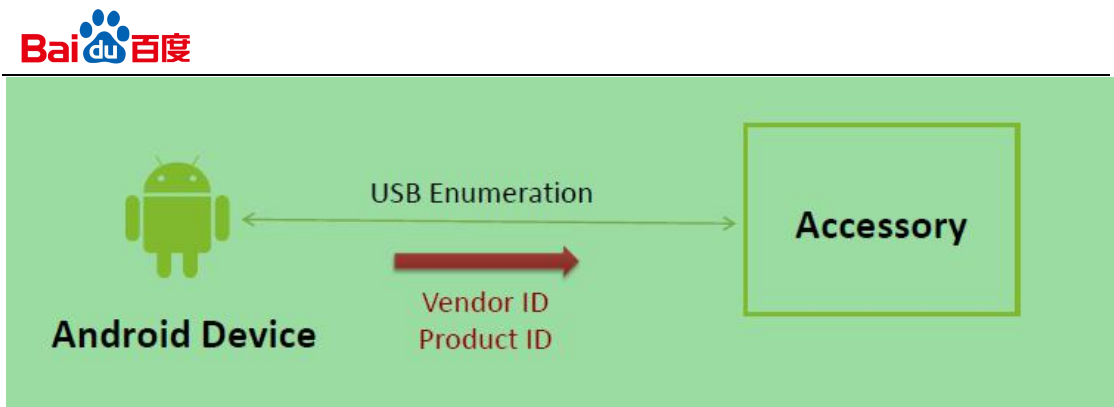


Figure 6.1-3: AOA Device Handshaking Process (2)

To judge if Android Mobile Device stays in Accessory Mode, the value of VID and PID should satisfy the following requirements:

Type	Value	Notes
VID	0X18D1	
PID	0X2D00	Supports Accessory Mode 1 interface with 2 bulk endpoints
	0X2D01	Supports Accessory Mode + ADB 2 interfaces with 2 bulk endpoints each
	0X2D02	Supports Audio 2 Audio interfaces (control + streaming)
	0X2D03	Supports Audio + ADB 3 interfaces: 2 Audio + 1 Bulk
	0X2D04	Supports AOA 1.0 + Audio 3 interfaces: 2 Audio + 1 Bulk
	0X2D05	Supports AOA 1.0 + Audio + ADB 4 interfaces: 2 Audio + 2 Bulk

Table 6.1-1: VID & PID Values of Andriod Mobile Device in Accessory Mode

3. The accessory sends out query command to Android Mobile Device. For most of Android Devices, under default circumstances, they do not mount Accessory drivers. That is to say, they stay in Accessory Mode by default. When the accessory establish USB connection with Android Mobile Device, the accessory will confirm if the Mobile Device is Android Device and if it supports AOA protocol or not as well as confirm the version number, through the

21 / 187



handshaking protocol. The accessory writes No.51 command to the port of endpoint 0. At this time, if the returned value is 1 or 2, it means the Android Mobile Device supports AOA1.0 or AOA2.0; but if the returned value is less than 1 or exceed 2, it means the Mobile Device does not support AOA protocol or it means the version number of the AOA it supports is not correct.

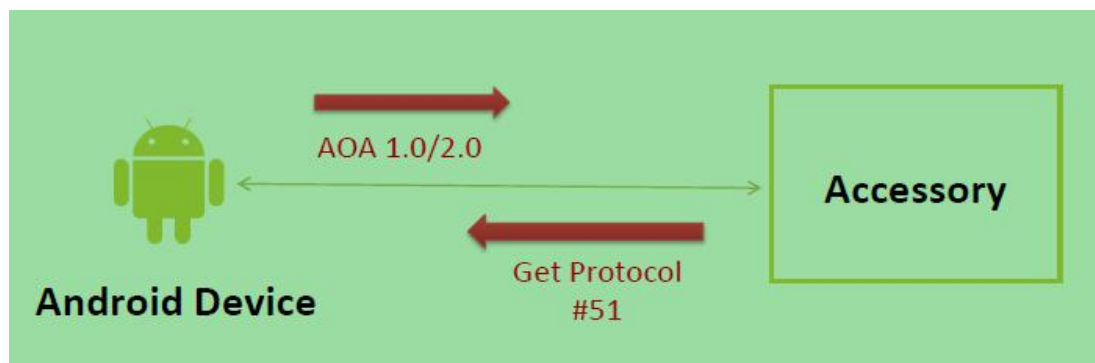


Figure6.1-4: AOA Device Handshaking Process (3)

4. Android accessory sends authentication information to Android Mobile Device, and sends out the command of starting communication.

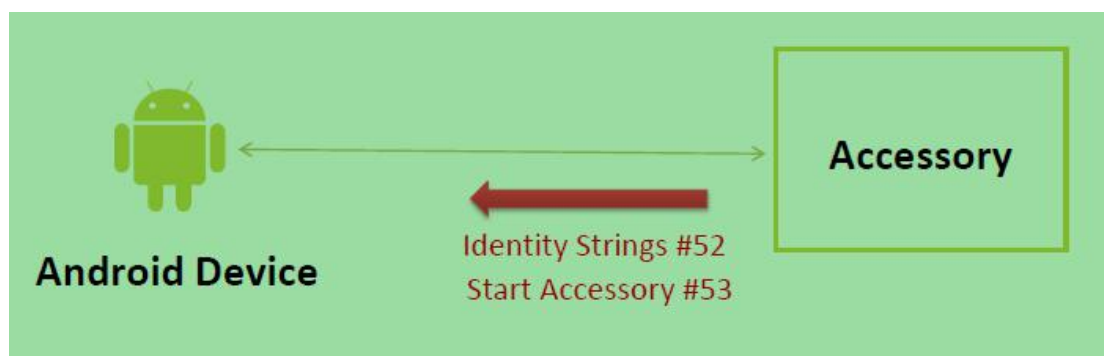


Figure6.1-5: AOA Device Handshaking Process (4)

Accessory writes in NO.52 command in Android Device, which includes the following messages:

No.	Content	Instance
1	Manufacturer	Baidu
2	Model Name	CarLife
3	Description	Baidu CarLife
4	Version	1.0.0



5	URL	http://carlife.baidu.com/
6	Serial Number	0720SerialNo.

Table 6.1-2: AOA Handshaking Additional Information

The accessory will send out No.53 command to Android Mobile Device, requesting Android Mobile Device to switch to AOA mode. Then Android Mobile Device will run this request and switch USB to AOA mode. During this process, there will appear a logic USB disconnection, and the accessory will again enumerate the Mobile Device, and then will follow step 2. After Android Mobile Device receives the parameter information in No.52 command sent from the accessory, it will use three parameters (Manufacturer, Model Name, and Version) to confirm the Mobile Device has been bound to the App of this Accessory. If there is no App within the system that can be matched with those three parameters mentioned above, the Android Mobile Device will pop up a dialog to offer description and URL information sent from Accessory for users. Thus, users are able to click URL to visit that webpage that has been directed. For details, please refer to the following figure:

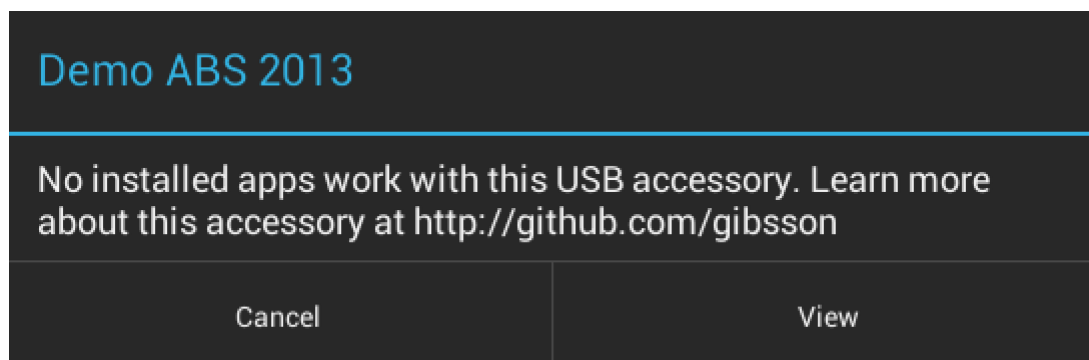


Figure6.1-6: AOA Device Handshaking Process (5)

If there is an App within the system that can be matched to the handshaking message sent from Accessory, then Android Mobile Device will pop up a dialog to ask users whether to immediately launch this App or not. If the user selects OK, this App will be turned on. Meanwhile, it will provide a checkbox. Once this box is ticked, Accessory will automatically start this APP each time. In fact, users are required to tick this box; or else, when this APP is started to request Accessory from USB Manager, it might not be able to turn on file descriptor or establish connection due to Permission problem. Details can be seen as below:

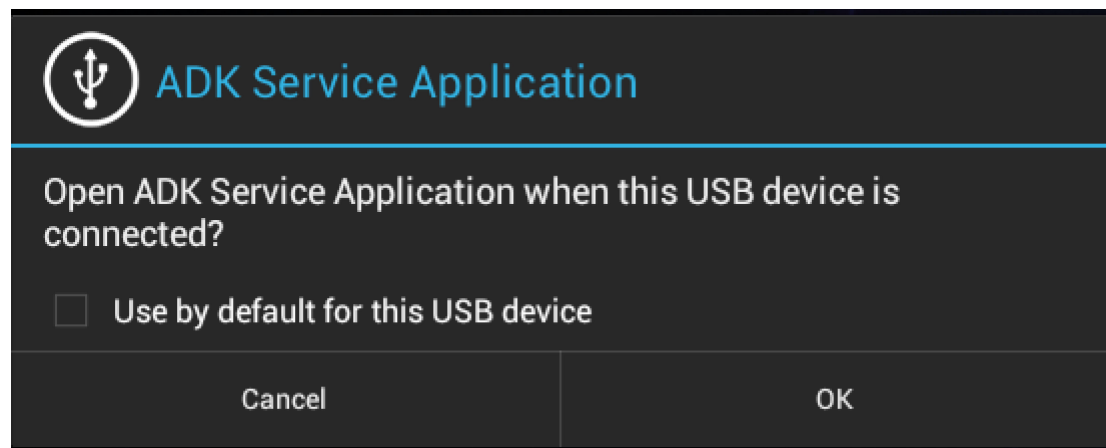


Figure6.1-7: AOA Device Handshaking Process (6)

5. Establish Connection

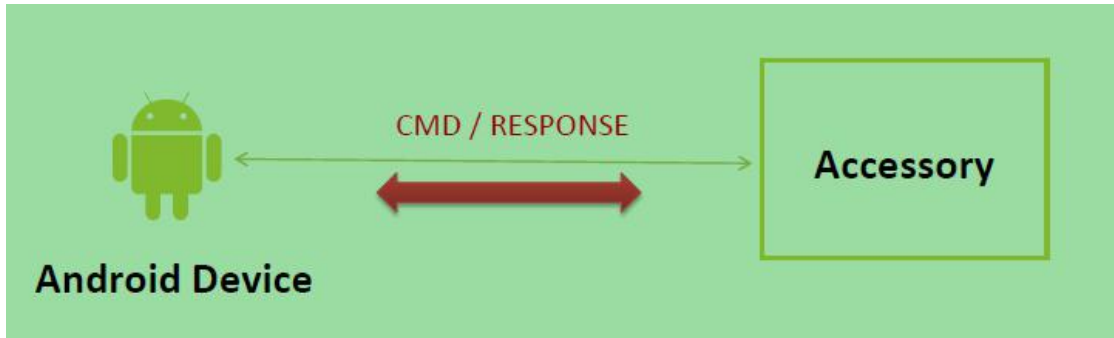


Figure6.1-8: AOA Device Handshaking Process (7)

6.1.1.2.2. AOA Device Handshaking Command

No.	Functions	Application in CarLife
51	To judge whether the accessory supports AOA protocol, and to attain the version number which supports AOA protocol.	Yes
52	To send out authentication information of the accessory, to help Android Mobile Device identify the App that is bound to the accessory.	Yes
53	Request Android Mobile Device to switch to	Yes



	Accessory Mode.	
54~57	HID Back-up	No
58	Audio Back-up	No

Table 6.1-3: AOA Device Handshaking Command

6.1.1.2.3. AOA Device Handshaking Flow

In conclusion, the handshaking flow of AOA device is displayed as below.

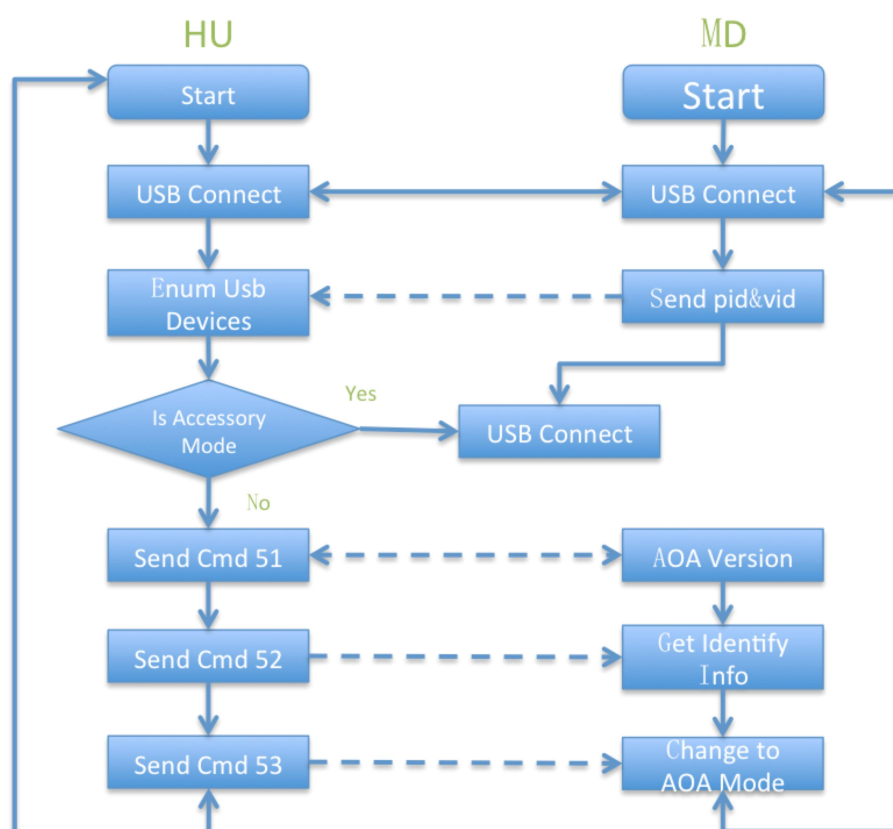


Figure 6.1-9: AOA Device Handshaking Flow Chart

6.1.1.3. AOA Message Structure

In order to not influence other modules (like audio, video, voice recognition or map modules), to reduce the workload as well as to develop compatibility in other connection methods, the message structure of each channel remains unchanged. (For details, please refer to Chapter 10 Communication Protocol. Since there is only one channel during AOA transmission process, a head need to be added based on previous message structure , so as to identify which



channel this message belongs to. Specific structure can be seen as below:



Figure 6.1-10: AOA Connection Package Structure

As this Figure shows, AOA Head takes up 8 bytes, while Channel ID and Msg Len respectively take up 4 bytes. For details, please refer to the following Table:

Table6.1-4: AOA Connection Package Structure Field Description

Field Name	Offset	Size(bit)	Description
Channel ID	0~3	4	It indicates which channel that this message belongs to. The channel of Cmd, Video, Media, TTS, VR, and Touch are respectively 1, 2, 3, 4, 5, and 6.
Msg Len	4~7	4	The total length of CarLife Msg
CarLife Msg	8~n	n-8	The message structure of CarLife Msg. Details can be seen Chapter 10 Communication Protocol.

When HU sends out AOA data, 8-byte AOA Head and CarLife Msg shall be sent separately.

According to 6.1.1.2, after No. 51, No.52 and No.53 commands run on HU, the HU will re-scan the USB accessory connecting to the HU, and the USB port that the Mobile Device reports to is AOA port, thus getting two objects (UsbDevice and UsbDeviceConnection). Input and output operation can be done via those two objects. CarLife program on MD will be automatically launched and UsbAccessory Object can be obtained, thus getting FileOutputStream and FileInputStream. Input and output operation can be done via those two objects. Details can be referred to in Android official documents.

6.1.1.4. AOA Development on HU

Currently, there are different HU systems such as Android, Linux, QNX and WinCE. On



Android system, Android officially offers USB interfaces for AOA connection, which mainly includes UsbManager, UsbDevice, UsbInterface, UsbDeviceConnection and so on. As for non-Android Head Unit without those interfaces mentioned above, open source lib like libusb should be used to directly operate USB device, including sending relevant command or transmitting data and so on. Detailed procedure is similar to the process of using interfaces offered by Android system.

6.1.1.5. AOA Data Channel Set-up

When setting up connection between MD and HU through AOA, it is recommended to establish data channel with MD through the following methods: set up socket local loopback structure on HU, that is to say, set up six socket communication for six channels. After reading in data from USB port via AOA and Libusb interfaces, unpack (remove AOA head, 8 bytes), and get CarLife Msg structure (For details, please refer to Chapter Communication Protocol), and CarLife Msg can be distributed to the according socket basing on Channel ID. Other modules (like Cmd, Video, Media) can read in data from socket and parse CarLife Msg to get concrete data. On the contrary, when other modules need to transmit data to MD, write in data in corresponding socket. After AOA connection module receives data via socket, add AOA head (Channel ID and Msg Len, 8 bytes in total), and call output interface to write in USB port and send to MD. Throug this way, it realizes message buffer mechanism via socket, and decouples the underlying module and CarLife upper module. Details are shown in Figure 6.1-11:

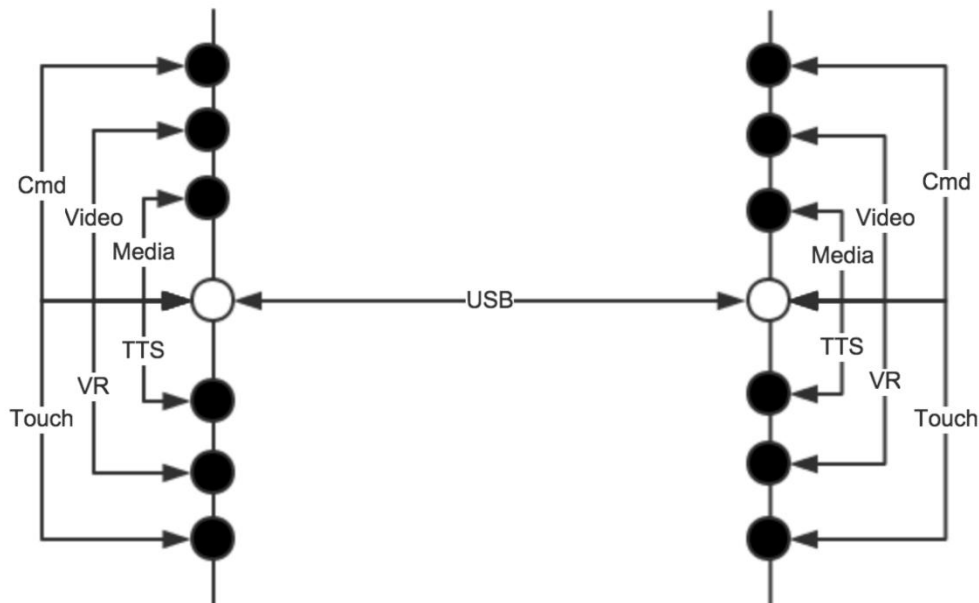




Figure 6.1-11: AOA Connection Channel Set-up

6.1.1.6. Definition of CarLife AOA Connection Animation Percentage on HU

During connecting of CarLife on HU, a connection animation is required (refer to the connecting in HMI section); for percentage of connection progress, it is recommended to make distribution in the following ways to facilitate positioning of user feedbacked issues:

Percentage (%)	Remark
10	After Android MD is detected
30	USB initialization (such as obtaining interface, etc.)
40	Detects whether it is in AOA mode
50	Switch to AOA mode
60	Get the two EPs (EndPoint) of AOA
80	Complete 6 socket connection of single channel simulation
100	6 socket connection is successful

Table 6.1-5: Percentage of AOA Connection

6.1.1.7. CarLife Launch & Identification on MD

6.1.1.7.1. CarLife Launch on MD

When using AOA connection, methods of launching CarLife on MD under different connection status can be seen in following table:

Connection Status	Running Status of CarLife on MD	Launch Method
Connecting	CarLife on MD running in background	AOA No.52 command
Connecting	Not running	AOA No.52 command
Connected	CarLife on MD running in background	Apply control message: MSG_CMD_GO_TO_FOREGROUND



6.1.1.7.2. CarLife Installation Status on MD

If CarLife has not been installed on MD, a dialog will pop up to notify users when AOA is connecting. Currently, there is still not a perfect way to judge whether CarLife has been installed on MD. One possible way is that if HU detects that no data is received in a while in AOA connection, HU will display a QR-code for downloading CarLife APP on MD:

(<http://webpage.navi.baidu.com/static/webpage/carlife/carlifedownload.html>)

This QR-code address will keep unchanged, but the downloading link will be updated periodically according to the version updating of CarLife APP on MD. When Android Phone or iPhone access this page, it will directly download the latest version of CarLife APP.

6.1.1.7.3 Method of Judging Android System Low Version With AOA Connection

HU should give a prompt when user is using a MD with Android system lower than 5.0. The way for the HU to judge Android system version is as below.

The HU connects and communicates with MD according to normal logic. When the HU receives the message (MSG_CMD_MD_INFO 0x00010004) from the MD, and identifies the current MD SDK less than 21, it can judge that the Android system of MD is lower than 5.0. At this time, the HU should disconnect the connection with the MD and give a prompt on HU screen with 'MD Android system lower than 5.0 is not supported'.

6.1.2. EAP Connection

EAP (External Accessory Protocol) is Apple's official communication protocol for communicating with external hardware connected to an iOS-based device. EAP connection mainly includes two methods: EA Session and EA Native Transport. EA Session is primarily used to transmit iAP2 command message, and the transmission rate is relatively low; EA Native Transport can report interface with two EPs, and the transmission rate is more efficient. CarLife EAP connection adopts the second method (EA Native Transport). This solution refers to Apple's official document *MFi Accessory Interface Specification R23*, and hereinafter referred to as R23.

6.1.2.1. Development on HU

The development of iPhone USB connection on HU mainly includes: MFI certification, USB drive, data reading and writing, and the establishment of data channels. MFI certification and USB are closely related to the hardware, which cannot be finished by the APP alone. Therefore, it is necessary for HU to meet certain hardware and software requirements. The

proposed development flow is shown as below:

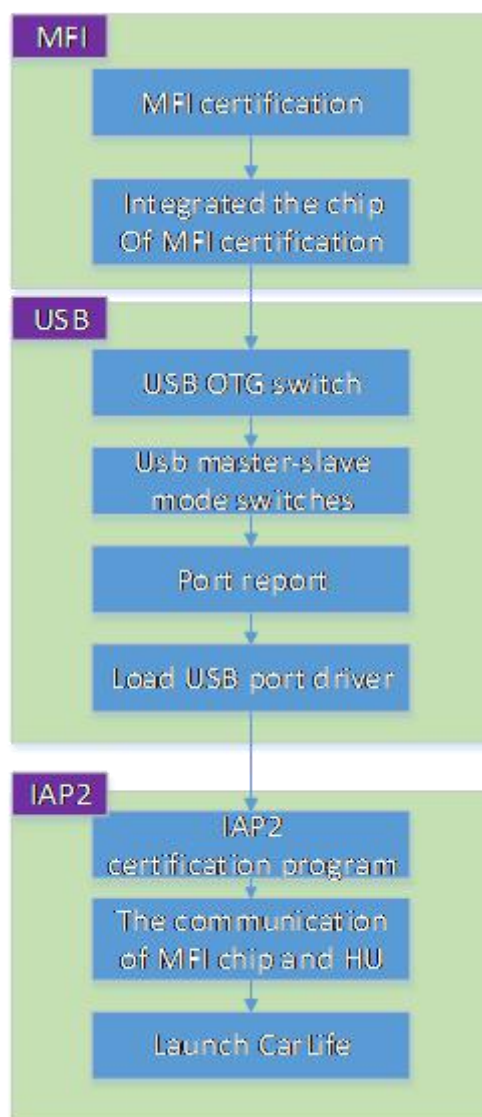


Figure 6.1-12: Development Flow on HU

1. MFI Certification

HU manufacturers need to assess MFI, and submit product plans on Apple MFi official website <mfi.apple.com> for MFi certification.

2. MFI Certified Chip Integration

After MFI certification is passed, purchase encryption chip from Apple, and integrate it onto HU.

3. OTG Switch

The USB port on Head Unit shall support OTG switch. When connecting with Android phone, HU stays at Host Mode; when connecting with iPhone, HU switches from Host Mode to Device Mode.



4. Host-device Mode Switch

When iPhone connects to HU, HU will act as HOST to enumerate iPhone. If the MD is detected as iPhone, HU will send role switch USB request to MD, waiting for iPhone to enumerate HU. After enumeration is finished, HU will act as Device.

5. Port Report

When HU switches from Host Mode to Device Mode, HU should report iAP2 and EA Native Transport port to MD. iAP2 is mainly used for data authentication while EA Native Transport port is mainly used to transmit data message of CarLife.

6. Load USB Port Driver

After HU reports iAP2 and EA Native Transport port to MD, it needs to load corresponding driver.

7. IAP2 Authentication

Before iPhone sets up data communication with HU, it needs to pass Apple authentication, that is, IAP2 authentication with MD via MFI certified chip according to R23. Only after authentication can CarLife data be transmitted through USB port.

8. Interaction between Certified Chip and HU Master Chip

MFI certified chip makes interaction with HU master chip through I²C bus. After loading the driver of MFI certified chip, the device file of MFI chip can be mapped in system device file, and IAP2 program set up communication with MFI chip through this device file.

9. Launch of CarLife

CarLife is launched through the Apple Protocol *App Launch*. Please refer to App Launch in R23.

6.1.2.2. USB Connection Layer Development

6.1.2.2.1 Connecting Process

1. The General Connecting Process:

The connection between iPhone and HU via USB is displayed in the following Sequence Diagram:

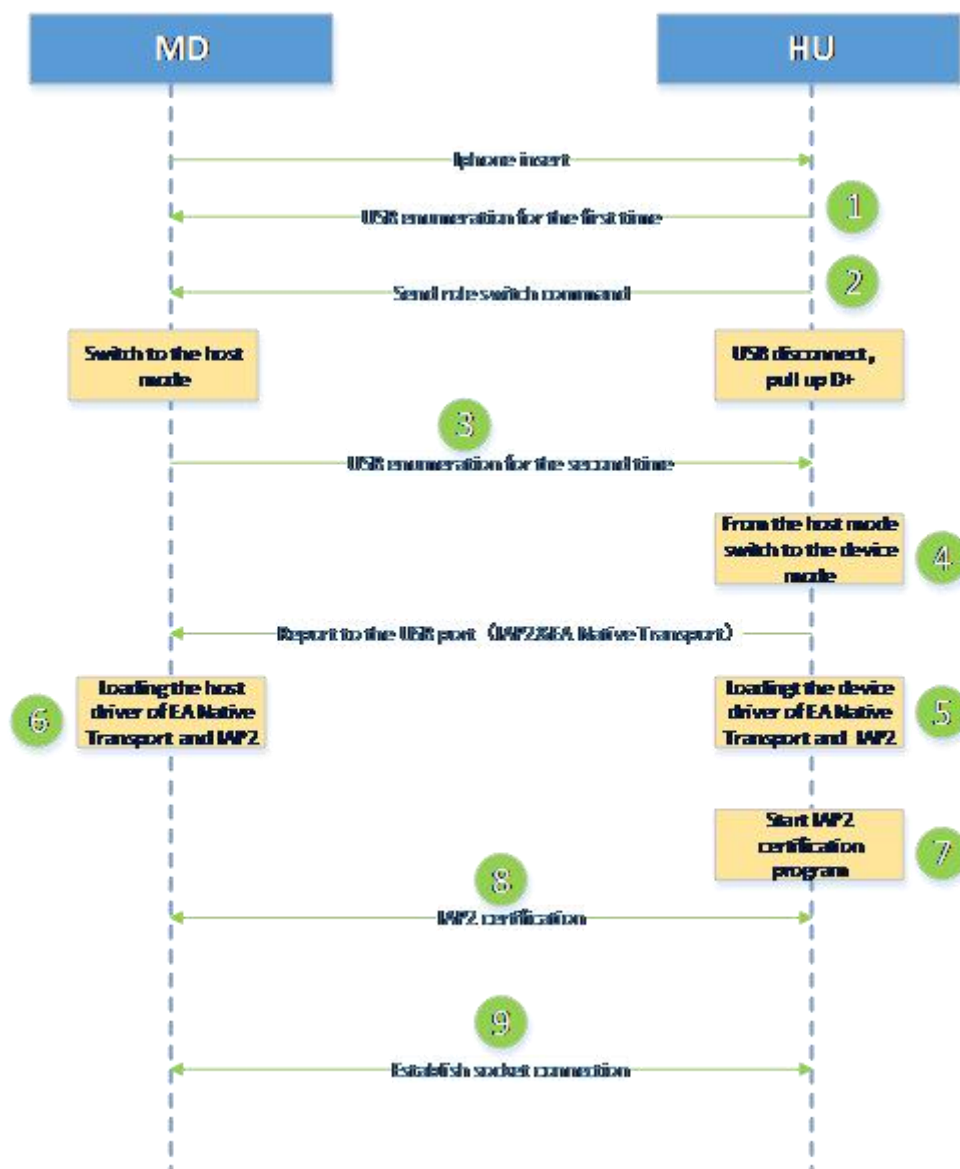


Figure 6.1-13: Connection Process Sequence Diagram

1. HU (as Host) Enumerates MD

When iPhone connects with HU, HU will act as HOST to enumerate MD. Through enumeration, HU can get the port info, and judge whether the connected MD is iPhone.

2. HU Sends Role Switch command

HU sends role switch command to MD via ep0 of MD, and disconnect USB connection. After MD receives the command, USB will reset and switch to HOST Mode.

3. MD (as HOST) Enumerates HU

When MD switches to HOST Mode, re-enumerate the HU (connected via USB).

4. HU Mode Switch

After enumerated by MD, HU will switch to Device Mode, and report to MD the standard EA



Native Transport port and IAP2 port conforming to Apple Authentication Protocol.

5. HU Loads Device Driver

HU loads EA Native Transport (mapped device file/dev/mfi-ean on HU), IAP2 device driver, and the driver of MFI certified chip (mainly interact with HU via I²C). After that, map relevant device file (/dev/MFi) under dev directory.

6. MD Loads HOST Driver

MD will load HOST driver of USB. (Apple has already finished this part)

7. MD Launches IAP2 Authentication Program

8. iAP2 Authentication

Develop authentication program according to Apple iAP2 Authentication Protocol. This program will interact with MD via /dev/MFi_iAP2 (the mapped device file of iAP2 Driver), and have data interaction with MFI certified chip via /dev/MFi (the mapped device file after MFI certified chip driver loading).

9. Establishment of Six Socket Channels

EA Native Transport port needs to be virtualized as six socket channels to establish six TCP Socket channels: command stream (CMD) two-way data transmission; video stream (Video) one-way data transmission, transmitted from MD to HU; audio stream (Audio) (2 TCP sockets, one for Media audio stream while the other for TTS audio stream, and both are one-way data transmission, sent from MD to HU); voice stream (VR), two-way data transmission, from HU to MD; back control stream (Touch), one-way data transmission sent from HU to MD.

2. EA Native Transport

When using EA Native Transport for transmission, MD should stay in Host Mode. Two interfaces reported from HU are determined by App on MD. When EA Session is turned on from APP, it needs to switch to a port with two EPs (bulk IN and OUT) for communication; when EA Session is turned off, it needs to switch to interface without EP. Here is an example of EA Native Transport for reference:



Figure 6.1-14: EA Native Transport Flowchart

6.1.2.2.2. Mode Switch

1. Mode Switch Process

After iPhone connects with HU, HU will detect and enumerate it. When the MD is detected

as an iPhone, HU will send a role switch command according to the following flowchart:

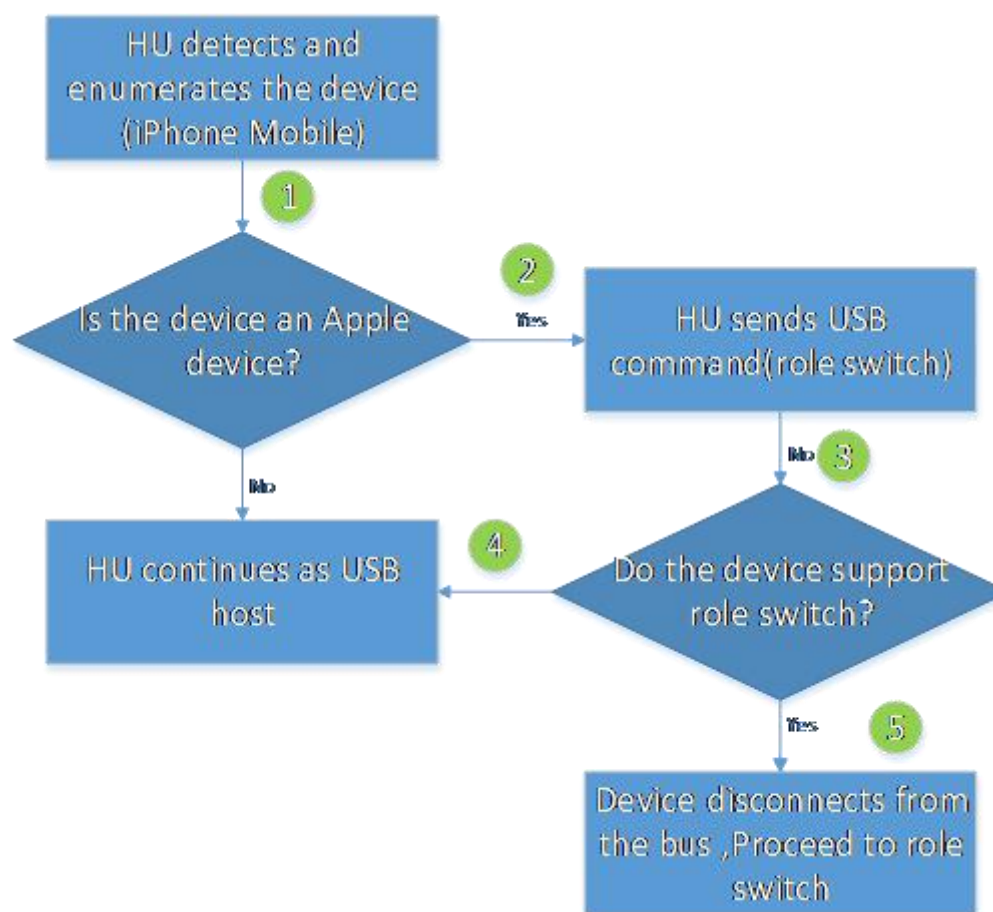


Figure 6.1-15: USB Role Switch Process

- (1) After the MD connects with HD, HD will determine MD type according to enumerated info (device VID, PID). If the MD is not an iPhone, HU will remain in HOST mode.
- (2) When an iPhone is detected, HU will send role switch command to the MD via EP0 to switch role.
- (3) After MD receives role switch command, it will determine whether it supports or not first.
- (4) If MD does not support role switch command, HU will not give any response.
- (5) If MD supports role switch command, HU will automatically disconnect USB connection with MD and launch D+ for MD to re-enumerate.

2. Mode Switch Command

When HU detects the connected MD as an iPhone, it will send the following command to iPhone via EP0 to switch iPhone's host-device mode, and the command format is shown as below:



Field	Value	Comments
bmRequest Type	0x40	Host to device request, vendor-defined type, device is recipient
bRequest	0x51	Vendor-defined USB role switch request
wValue	0x 00	
wIndex	0x00	Reserved
wLength	0x00	No data transfer

Table 6.1-6: Mode Switch Command

6.1.2.2.3. Reported USB Port

When HU switches to Device Mode, it needs to report two USB ports to MD: EA Native Transport port and iAP2 port, which are respectively used for data transmission and IAP2 certification.

1. EA Native Transport Port

EA native transport has two interfaces: one interface is with two EPs, namely bulk IN and bulk OUT (This interface should be used for data transmission); the other interface has no EP, which can not be used for data transmission. Port attributes are as follows:

USB Descriptor	Value	Comments
Interface Number	0xNN	Must be different from the iAP2 interface number
Interface Class	0xFF	Vendor-specific interface
Interface SubClass	0xf0	Mfi Accessory
Interface Protocol	0x01	External Accessory Native Transport
Interface String	com.baidu.CarLifeVehicleProtocol	Must match an identified External Accessory Protocol Name
Number of Endpoints	2	one bulk In and one bulk OUT endpoint
Alternate Setting	1	

Table 6.1-7: EA Native Transport Alternate Setting 1

USB Descriptor	Value	Comments
Interface Number	0xNN	Must be different from the iAP2 interface number
Interface Class	0xFF	Vendor-specific interface
Interface SubClass	0xf0	Mfi Accessory
Interface Protocol	0x01	External Accessory Native Transport
Interface String	com.baidu.CarLifeVehiclePro	Must match an identified External



	toicol	Accessory Protocol Name
Number of Endpoints	0	one bulk In and one bulk OUT endpoint
Alternate Setting	0	

Table 6.1-8: EA Native Transport Alternate Setting 0

2. iAP2 Port

If HU needs to set up iAP2 communication with iPhone (in Host Mode), it must report a special USB port (two EPs: bulk out & bulk in) port, namely iAP2 port.

USB Descriptor	Value	Comments
Interface Number	0x00	
Interface Class	0xFF	Vendor-specific interface
Interface SubClass	0xf0	Mfi accessory
Interface Protocol	0x00	
Interface String	iAP Interface	
Number of Endpoints	2	one bulk In and one bulk OUT endpoint

Table 6.1-9: iAP2 Port Info

iPhone (in Host Mode) sends data to HU via bulk out EP. If HU receives data correctly, it needs to reply a USB ACK package to iPhone HOST port; if data is in error or HU is not ready, HU needs to reply a USB NAK package.

6.1.2.3. Authentication

Communication between HU and iPhone requires authentication via certified chip. The current versions of certified chip are 2.0B and 2.0C. 2.0C Integration is the officially recommended version. The interaction flow of MD, HU and certified chip is shown as below:

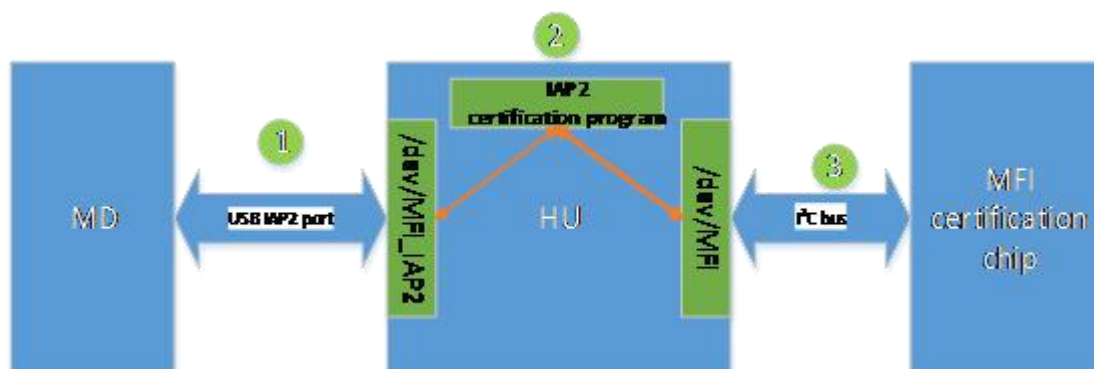


Figure 6.1-16: Authentication Process

Authentication command between MD and HU is transmitted via iAP2 port, which is reported



to MD when HU switches to Device Mode. HU will start iAP2 certification program after USB driver loading is completed. This program communicates with MD via the mapped device file of iAP2 port (/dev/MFi_iAP2), and communicates with MFi certified chip via the mapped device file during MFi certified chip driver loading process. MFi certified chip connects with HU via I²C bus, and the transmission conforms to I²C protocol.

6.1.2.3.1. Accessory Authentication

iAP2 certification program runs according to Apple official file R23 (Chapter 12 Accessory Authentication). The involved authentication certificate messages are as follows:

1. RequestAuthenticationCertificate

Command parameters are shown as below:

Source	ID	Name	ID	Type	#	Notes
Device	0xAA00	This message has no parameters				

Table 6.1-10: RequestAuthenticationCertificate Command

Accessory (HU) needs to respond to RequestAuthenticationCertificate command in one second. Detailed responses are listed in Command 2.

2. AuthenticationCertificate

Command parameters are shown as below:

Source	ID	Name	ID	Type	#	Notes
Accessory	0xAA01	AuthenticationCertificate	0	blob	1	Accessory's X.509 certificate

Table 6.1-11: AuthenticationCertificate

If the certificate is valid, Apple device will continue to send RequestAuthenticationChallengeResponse, and Accessory (HU) needs to calculate the response to Authentication Challenge via authentication chip, and send message to Apple device in two seconds. Details can be referred to Command 3 and Command 4.

3. RequestAuthenticationChallengeResponse

Command Parameters are shown as below:

Source	ID	Name	ID	Type	#	Notes
Device	0xAA02	AuthenticationChallenge	0	blob	1	Random#

Table 6.1-12: RequestAuthenticationChallengeResponse

4. AuthenticationResponse

Command Parameters are shown as below:

Source	ID	Name	ID	Type	#	Notes
Accessory	0xAA03	AuthenticationResponse	0	blob	1	Computed challenge



						response
--	--	--	--	--	--	----------

Table 6.1-13: AuthenticationResponse Command

5. AuthenticationFailed

Command Parameters are shown as below:

Source	ID	Name	ID	Type	#	Notes
Device	0xAA04	This message has no parameters				

Table 6.1-14: AuthenticationFailed Command

6. AuthenticationSucceeded

Command Parameters are shown as below:

Source	ID	Name	ID	Type	#	Notes
Device	0xAA05	This message has no parameters				

Table 6.1-15: AuthenticationSucceeded Command

Apart from those commands mentioned above, such commands as StartIdentification, IdentificationInformation, IdentificationAccepted, IdentificationRejected, CancelIdentification, and IdentificationInformationUpdate will also be used. Details can be referred to Chapter Accessory Identification in Apple's official file.

When HU receives authentication commands sent from iPhone, it will send a request to MFi certified chip, and MFi certified chip will react according to specific command algorithm, and return the result to iAP2 certification program on HU, and then to iPhone. The following Sequence Diagram shows the process of a successful authentication:



After device authentication mentioned in Section 3.1, if it needs to support EAP function, identification depicted in this section should be done. Details can be referred to chapters of MFI certification document (R23, please refer to Chapter 13 Accessory Identification and Chapter 59 iAP2 Control Session Messages).

40 / 187



Figure 6.1-18: Accessory Identification Sequence Diagram

The key here lies in the parameters of IdentificationInformation command. Please refer to (Section 59.2.2 IdentificationInformation in R23) for details. Particularly, SupportedExternalAccessoryProtocol (Table 59-11 in R23) and USBHostTransportComponent (Table 59-15 in R23) are extremely important parameters. ExternalAccessoryProtocolName in SupportedExternalAccessoryProtocol command is from *com.baidu.CarLifeVehicleProtocol*.

As for the identification process between Apple device and accessory, please refer to Chapter 12 Accessory Authentication, Chapter 13 Accessory Identification and Chapter 59 iAP2 Control Session Messages in R23.

6.1.2.4. Launch CarLife on MD

The launch of CarLife on MD is finished via App Launch in Apple MFI Authentication Protocol. CarLife on iPhone can be launched via running APP Launch command to bind Bundle ID of CarLife. Detailed command is shown as below:

Source	ID	Name	ID	Type	#	Notes
Accessory	0Xea02	AppBundleID	0	Utf8	1	com.baidu.lbs.CarLife
		LaunchAlert	1	enum	0/1	See Table 8-1-1

Table 6.1-16: App Launch Command

The values of LaunchAlert are shown as follows:

value	Meaning
0	Launch with user alert (default)
1	Launch without any alert, but device must be unlocked and on Home



	Screen or Music/Video app.
--	----------------------------

Table 6.1-17: AppLaunch Method enum

Main scenarios for launching CarLife are shown as follows:

Connection Status	Running Status of CarLife on MD	Launch Command
Connecting	CarLife on MD running in background	App launch
Connecting	Not running	App launch
Connected	CarLife on MD running in background	App launch

Table 6.1-18 Main Scenarios for Launching CarLife on MD

Launch with user alert is highly recommended by Baidu. Launch without any alert requires that the MD is not locked and on Home Screen or Music/Video App; or else, launch will have no response. Launch with user alert will have pop-up alert in any case.

6.1.2.5. EAP Connection Message Structure

In order not to influence other modules (like audio and video, voice, maps, and other modules), and to minimize the workload of code development as well as to ensure the compatibility with other connection methods, the message structure of EAP connection channels remains unchanged (Please refer to *Baidu CarLife Integration Data Communication Protocol* for details). Since only one channel is used in data transmission of EAP connection, an EAP Head needs to be added to label channel details based on the original message structure. Specific structure is shown as follows:

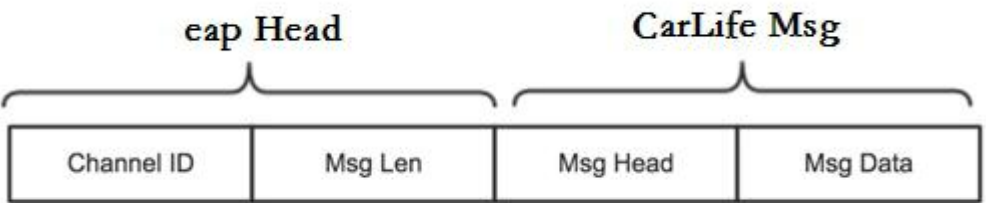


Figure 6.1-19: Message Structure of EAP Connection

As shown above, EAP Head has 8 bytes in total while Channel ID and Msg Len have 4 bytes respectively, which is shown in the following table:

Field Name	One-Bit Offset	Siz(bit)	Decription



Channel ID	0~3	4	To identify which channel this message belongs to. The channel No. Of Cmd, Video, Media, TTS, VR and Touch are respectively 1,2,3,4,5,6
Msg Len	4~7	4	The overall length of CarLife Msg
Msg Head	8~m+7	m	The lengths of channels are different. Cmd and Touch have 8 bytes respectively while other channels have 12 bytes respectively.
Msg Data	m+8~n	n-m-8	The data types of channels are different. The data type of Cmd and Touch is serialized pb structure while the data type of Video channel is video stream, and the type of Media is PCM stream.

Table 6.1-19: Message Structure Description

6.1.2.6. Interaction Protocol of HU Driver Layer & CarLife on HU

HU Driver Layer requires synchronizing its status to HU CarLife Application Layer during NCM driver loading and iAP2 certification process. The purposes are as follows: firstly, CarLife requires the status of HU Driver Layer to display connection progress; secondly, HU Driver Layer could respond its status change timely to CarLife when plugging in or plugging out MD, in order to handle launching & disconnecting CarLife.

6.1.2.6.1. Communication Mode between HU Driver and CarLife on HU

Multiple communication modes between HU Driver and CarLife on HU can be chosen according to HU system features, such as local socket, broadcast, etc. HU manufactures can choose communication mode according to their targeted HU system. More details regarding to local socket communication mode will be introduced as follows:

1. The address for local socket communication mode is /data/local/tmp/ncm.sock
2. socketName = "/data/local/tmp/ncm.sock"
3. CarLife on HU serves as client while HU Driver Layer (iAP2 Certification Program) serves as Server



6.1.2.6.2. Messages Sent from HU Driver to CarLife on HU

HU Driver informs the progress of USB connection to CarLife on HU via different messages during different connecting periods. Detailed commands and descriptions are as follows:

Status	Value	Applicable Connection Mode	Comments
<i>sendUsbRoleSwitch</i>	0x01	EAP/NCM	When HU Driver sends RoleSwitch command will associate with this status
<i>iap2AuthBefore</i>	0x02	EAP/NCM	Before conducting iAP2 certification
<i>iap2AuthAfter</i>	0x03	EAP/NCM	After iAP2 certification completed
<i>allocationNcmIp</i>	0x04	NCM	After allocating IP successfully
<i>eapFileReady</i>	0x05	EAP	Enable read-write for mapping device nodes in EAP device driver
<i>usbDisconnect</i>	0x10	EAP/NCM	After plugging out iPhone

Table 6.1-20: Messages Sent from HU Driver to CarLife on HU

6.1.2.6.3. Messages Sent from CarLife on HU to HU Driver

The launch command is the only command that CarLife on HU can send to HU driver. Since CarLife on HU cannot launch CarLife on MD independently, it relies on App Launch command from Apple MFi Certification protocol. The App Launch command runs in iAP2 certification program. When HU needs to launch CarLife on MD, CarLife on HU needs to send the command for informing iAP2 certification program. After iAP2 program receives this command, it will send App Launch command via iAP2 port to launch CarLife on MD.

Status	Value	Comments
<i>pullUpCarLife</i>	0x11	Launch CarLife

Table 6.1-21: Messages Sent from CarLife on HU to HU Driver CarLife

6.1.2.7. Heartbeat Message from HU to MD

Unlike port-to-port connection of internet socket mode (no matter which side shuts off the communication, the other side can immediately know), regarding EAP connection, if the USB cable stays connected, when CarLife exits on HU, MD will not know. Therefore, it is necessary to add heartbeat mechanism between HU and MD. After HU receives MSG_CMD_PROTOCOL_VERSION_MATCH_STATUS, it needs to send a heartbeat message to MD, one message per second. It is the same as the heartbeat message (MSG_VIDEO_HEARTBEAT = 0x00020002) via video channel from MD to HU. MD will determine the activation status of HU via heartbeat message in EAP connection. If MD has not received the heartbeat message for more than **3 seconds**, it will disconnect the connection.

6.1.2.8. Methods of CarLife on HU Reading Messages from EA Channels

2 points needs to be followed for reading messages in EAP connection to HU:

1. In stead of relying on buffer of EA driver, HU upper layer must have its self-managed data buffer. After EA driver receives messages sent from MD, all messages shall be read into the data buffer of HU upper layer. This is because buffer of EA driver is quite limited; once getting full, new messages will override the previous messages, resulting in data dislocation, then failure of data parsing and then disconnection between HU and MD.

2. HU shall start a specific data parsing thread in CarLife App layer, which reads and parses data from HU buffer. HU buffer needs to be locked to ensure the multithreading safety. As for the parsing process, read EAP HEAD first, obtain MsgLen, and then read the length of MsgLen; if the existing data in buffer is less than MsgLen, wait until data equals MsgLen and then read the data; parse CarLife Msg then and loop read the next message.

6.1.2.9. Definition of HU CarLife EAP Connection Animation Percentage

During connecting of CarLife on HU, a connection animation is required (refer to the connecting in HMI section); for percentage of connection progress, it is recommended to make distribution in the following ways to facilitate positioning of user feedbacked issues:

Percentage (%)	Description
10	Sending during HU driver sending RoleSwitch command
30	Before IAP2 certification
60	After IAP2 certification
80	EAP device driver mapped device nodes can open read and write
100	6 socket connection is successful

Table 6.1-22: Animation Percentage of EAP Connection

6.1.2.10. Difference between CarLife and CarPlay

CarLife on HU and MD adopts EAP connection to transmit data while CarPlay adopts NCM



ports. Besides, CarLife and CarPlay are different in authentication, driver layer development, and connection as follows:

1. Authentication

MFfi authentication in CarLife needs to support IAP2, but for CarPlay, it also needs to get CarPlay official authentication.

2. Driver Layer

In driver layer, there lie differences in switching role switch commands in driver layer. Role switch commands in CarLife can be referred to 6.1.3.2.2. Role switch commands in CarPlay are shown as below:

Field	value	comments
bmRequest Type	0x40	Host to device request, vendor-defined type, device is recipient
bRequest	0x51	Vendor-defined USB role switch request
wValue	0x 01	
wIndex	0x00	Reserved
wLength	0x00	No data transfer

Table 6.1-23: Role Switch Commands in CarPlay

The primary difference lies in wValue. For CarLife, wValue is 0 (only for switching NCM port), but for CarPlay wValue is 1 (when switching to NCM port, it will enable MD to launch CarPlay).

3. Connection

EAP connection of CarLife visualizes one USB channel to six socket channels for communication while CarPlay relies on IP-based socket connection. CarLife has TCP socket channels while CarPlay has UDP socket channels. Besides, CarLife supports both IPV4 and IPV6 communication, but CarPlay only supports IPV6 communication.

6.1.2.11. Launch Strategy When CarLife Coexist with CarPlay on HU

Apple has defined the following standard rules for automatic launch of CarPlay upon USB connection in the latest *Accessory Interface Specification*, Chapter 12.2.4 *CarPlay over USB*.

- "The accessory must establish the CarPlay automatically upon physical connection to the Apple device."*
- "Once a connection has been established, setup and content transfer will start after the accessory completes authentication over CarPlay interface."*
- "The accessory must be able to establish a CarPlay session within 3 seconds of Apple device connection."*

Considering the coexistence between CarLife and CarPlay on HU, without violating Apple's launch standard for CarPlay, we suggest adopting the following launch strategy for CarLife



and CarPlay in iPhone connection. And this strategy has already been put into use and released to the market by one of our OEM partners.

First iPhone Connection

- 1) For CarPlay
 - HU will pop up with 2 selections asking which way users would like to launch CarPlay, manually or automatically.
 - If users choose 'Automatically' in this pop-up, then CarPlay will launch not only in this time, but also in every following connection automatically.
 - If users choose 'Manually', then CarPlay will stay on the HU page; for following connections, users are required to choose CarPlay manually on HU menu.
- 2) For CarLife
 - When users select CarLife button on HU menu for the first time, HU will also show a pop-up asking for CarLife's launch preference, manually or automatically.
 - If users choose 'Automatically' in this pop-up, then CarLife will launch not only in this time, but also in every following connection automatically.
 - If users choose 'Manually', then CarLife will stay on the HU page; for following connections, users are required to choose CarLife manually on HU menu.

Second iPhone Connection

- HU will judge whether users has selected specific launch preference previously.
- HU will launch the application according to users' selection in previous setting, no alert will pop up.
- HU will pop up the alert if launch preference hasn't been selected before.

Change Preference

- HU shall have the user accessible settings to allow users to set or change their preference for launching CarPlay and CarLife.

For details, please check the following flow chart:

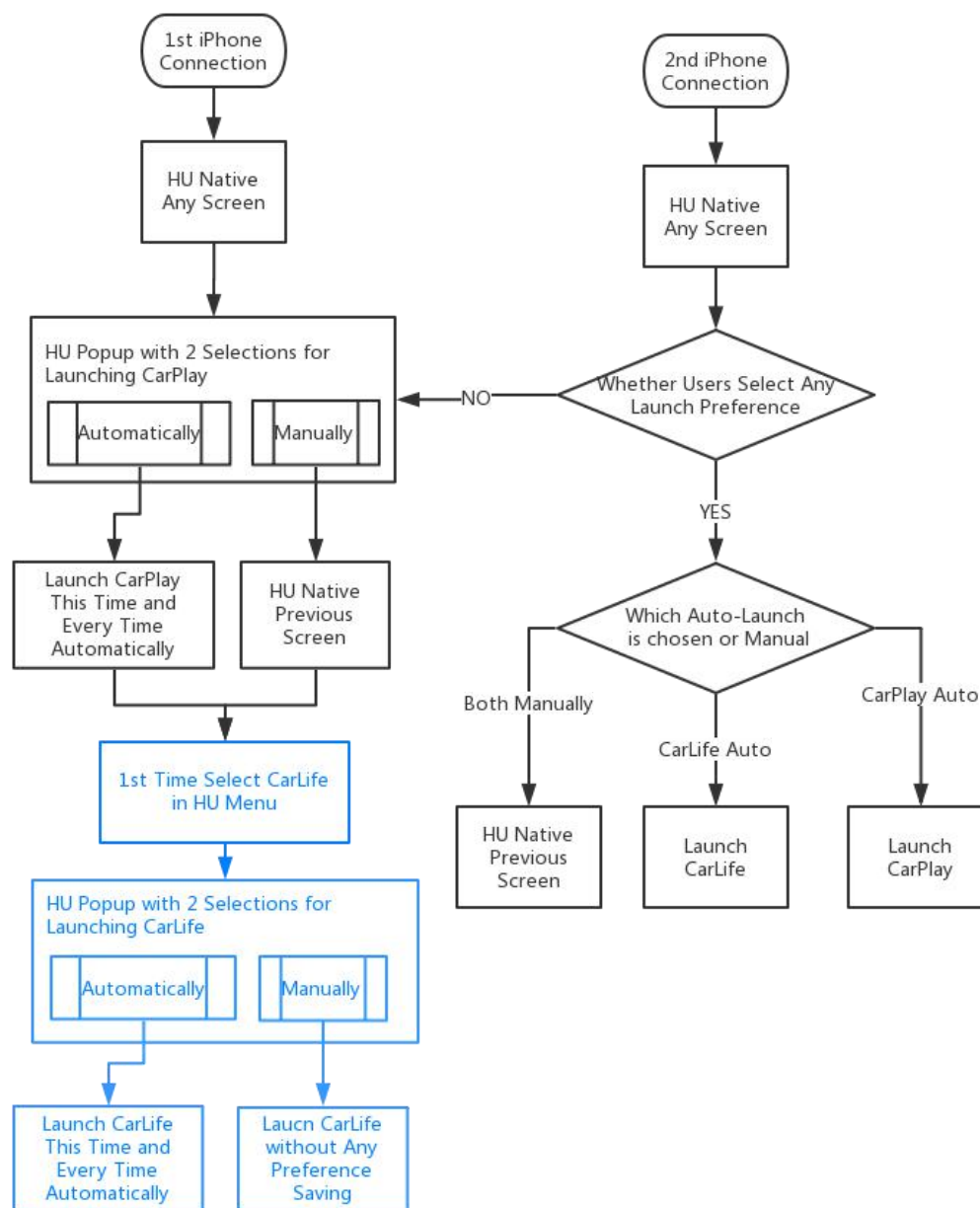


Figure 6.1-20 Launch Strategy for CarLife and CarPlay

6.1.2.12. CarLife MFi Certification Process

For supporting iPhone Lightning Connection, HU manufactures need to pass MFi certification. The following paragraphs mainly introduce several items when submitting MFi certification on mfi.apple.com.

(1) Accessory Overview

For technology, please tick iAP

Baidu CarLife (DuerOS Classic) Integration Specification



PFID: 113308-0004	Accessory Status: Active	Select Action ▾
Accessory Name: DISPLAY AUDIO	Process Phase: Production	
Base Model/SKU: J16DAW08A010	Accessory Type: Auto Aftermarket	

Status

Product Plan Form

Accessory Overview

Firmware & Hardware

Licensed Components

Manufacturing Info

Upload Sketch

Accessory Overview

Technology

☒ iAP

☐ CarPlay

Components

☒ Authentication coprocessor

☐ Lightning connector

☐ 30-pin connector

Auto Accessory Information

Will this accessory be a factory install/factory option from the auto manufacturer?*

☐ Yes

☒ No

Accessory name*

An Apple trademark cannot be part of your accessory name.

DISPLAY AUDIO

Will this be a new accessory or will it be identical to an accessory covered under another Product Plan?*

☒ New accessory

☐ Identical to another accessory

Please describe the accessory, including a functional overview, key features and how it will connect to an Apple device.*

It supports navigation, music play, photo browser, radio

Which category best describes your accessory hardware? Select one.*

☒ Head unit

☐ Vehicle Diagnostics

☐ Head unit with adapter or cable with microprocessor

☐ Other

Model Information*

(2) Firmware& Hardware

For iAP protocol, please tick iAP2, and it needs to support USB Host Mode and USB Device Mode. iAP2 shall support App Launch and EA Native Transport.

Status

Product Plan Form

Accessory Overview

Firmware & Hardware

Licensed Components

Manufacturing Info

Upload Sketch

Firmware & Hardware

Which iAP protocol(s) will the accessory implement? Select one.*

☒ iAP2

☐ iAP2 and iAP1

☐ iAP1

"iAP1" can no longer be selected. As stated in the technical specifications, all new accessories that implement iAP must support iAP2.

Which transport(s) will be used for iAP communication between the accessory and the Apple device?*

☒ USB Host Mode

☒ USB Device Mode

☐ Serial

☐ Bluetooth

Which of the following connectors will be integrated into your accessory, including any supplied cables or adapters? Select all that apply.

☒ USB Standard-A receptacle

☐ USB Micro-A plug

☐ USB Mini-B receptacle or plug

☐ 3.5mm headset jack

☐ USB Standard-A plug

☐ USB Standard-B receptacle or plug

☐ USB Micro-B receptacle or plug

☐ Other non-USB

Select the iAP features which will be used in this accessory.*

iAP2 Features

☒ App Launch

☐ Assistive Touch

☐ Bluetooth Connection

☐ Call Controls

☐ Call State Updates

☐ Communications Updates

☐ Device Authentication

☐ Device Notifications

☐ Device Powered

For supported phone models, please tick iPhone5 or beyond.

Baidu CarLife (DuerOS Classic) Integration Specification



Accessory Overview **Firmware & Hardware** Licensed Components Manufacturing Info Upload Sketch

Now playing

- ☐ Telephony
- ☐ Vehicle status
- ☐ VoiceOver
- ☐ Wi-Fi Information Sharing

Select any additional feature(s) that will be supported by the accessory. Select all that apply.

- ☐ Siri Eyes Free

Please specify the models which will be supported by your accessory.*
You will be required to complete self-certification for each supported model.

☒ iPhone 6s Plus	☒ iPhone 6s
☒ iPhone 6 Plus	☒ iPhone 6
☒ iPhone 5s	☒ iPhone 5c
☒ iPhone 5	☐ iPhone 4s
☐ iPhone 4	☐ iPhone 3GS
☐ iPhone 3G	☐ iPhone
☐ iPad Pro	☐ iPad mini 4
☐ iPad Air 2	☐ iPad mini 3
☐ iPad Air	☐ iPad mini 2
☐ iPad mini	☐ iPad (4th generation)
☐ iPad (3rd generation)	☐ iPad 2
☐ iPad	☐ iPod touch (6th generation)
☐ iPod touch (5th generation)	☐ iPod touch (4th generation)
☐ iPod touch (3rd generation)	☐ iPod shuffle (4th generation)
☐ iPod shuffle (3rd generation)	☐ iPod shuffle (2nd generation)
☐ iPod shuffle (1st generation)	☐ iPod classic
☐ iPod touch (2nd generation)	☐ iPod touch (1st generation)
☐ iPod nano (7th generation)	☐ iPod nano (6th generation)
☐ iPod nano (5th generation)	☐ iPod nano (4th generation)
☐ iPod nano (3rd generation)	☐ iPod nano (2nd generation)
☐ iPod nano (1st generation)	

(3) Certified Chip

For certified chip, please tick MFI337S3959-2.0C.

PPID: 113308-0004 Accessory Status: Active
Accessory Name: DISPLAY AUDIO Process Phase: Production
Base Model/SKU: J16DANDHA010 Accessory Type: Auto Aftermarket

Select Action

Accessory Overview **Firmware & Hardware** **Licensed Components** Manufacturing Info Upload Sketch

Licensed Components

Authentication Chips*

- ☐ MFI341S2162 - Class 4, 2.0B
- ☐ [Auto] MFI341S2313 - Class 4, 2.0B WTR
- ☒ MFI337S3959 - 2.0C

(4) CarLife Info

CarLife bundleid: com.baidu.lbs.CarLife This bundleid can be used to launch CarLife.



6.2. Bluetooth Auto-pair (HU Can Choose Whether to Support this Function)

Bluetooth auto-pair means that after **HU and Android phone are successfully connected, they can finish interaction via specific protocols** to achieve Bluetooth auto-pair and connection. During this process, user's confirmation or typing is not needed. Meanwhile, Bluetooth telephone feature will be activated only when Bluetooth connection has been established. (iOS system doesn't support BT auto-pair currently)

There are several points regarding BT auto-pair worth mentioning:

- Due to the limitation of iOS system, it currently does not support BT auto-pair, and users need to connect Bluetooth manually;
- **After Android 6.0, the BT address is obtained by reflection as it is unable to be obtained from official interface, which cannot guarantee its correctness. Currently, Google is becoming more strict on the access to BT address on Android MD. Thus it is suggested that HU does not support BT automatic pairing. The MD CarLife will not support BT pairing in Android 9.0 and above, while the lower version remains unchanged.**
- BT auto-pair function requires certain modification on HU, which mainly is, auto-acceptance of BT connection request from MD;
- Turning on BT auto-pair function needs to respond the configuration in MSG_CMD_HU_FEATURE_CONFIG_RESPONSE. Details can be referred to in the definitions of MSG_CMD_HU_FEATURE_CONFIG_RESPONSE
- BT auto-pair function is optional for OEM partners. Partners could choose to support or not support this function based on their own conditions.
- If HU has ever connected to other Bluetooth device before, it needs to disconnect with the previous device to connect with CarLife after turning on BT auto-pair.

6.2.1. BT Auto-pair Mode

Bluetooth has various pairing methods. It depends on Bluetooth version and devices' input/output ability to choose the pairing method. In the age of Bluetooth 2.0 and before, traditional PIN CODE mode is the only choice, but after Bluetooth 2.1 comes out, a new Bluetooth pairing method has been developed call Simple Secure Pairing (aka. SSP).

Currently CarLife' auto-pair in Bluetooth will apply to traditional PIN CODE pairing and SSP's PASS KEY comparison pairing method.



6.2.1.1. PIN CODE Mode

In the age of Bluetooth 2.0 and before, it is the only way for Bluetooth pairing. Both sides of pairing should input pre-set PIN CODE for pairing; wrong PIN CODE from any side will result in pairing failure.

6.2.1.2. SSP Mode

With Bluetooth 2.1 version introducing SPP paring mode, more pairing methods could be used. Currently CarLife support PASS KEY comparison paring mode; other paring modes such as OOB, PASS KEY Entry, Just Work mode are not supported in CarLife

As regards to PASS KEY’s comparison method, both sides for paring only need to compare 6 digits numbers PASS KEY to accept or refuse paring. If numbers match, users can click ‘confirm’ to accept paring, otherwise refuse pairing.

6.2.2. Auto-pair Sequence

Before starting auto-pair, HU must know whether MD supports BT auto-pair via MSG_CMD_HU_FEATURE_CONFIG_RESPONSE. If MD hasn’t received FEATURE_CONFIG from HU, auto-pair will not be successful.

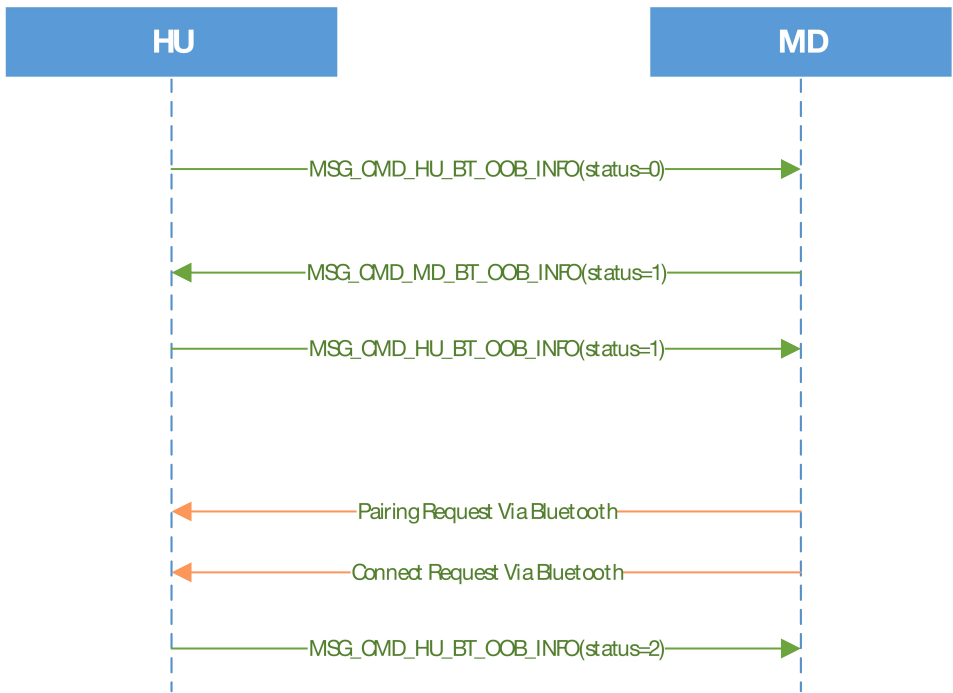


Figure 6.2-1
After CarLife is connected, HU will send message



MSG_CMD_BT_START_IDENTIFY_REQ to MD. When MD receives this message, it will judge whether the HFP connection between HU and MD is established. If the connection is established, auto pairing won't be activated (iOS doesn't support auto pairing). If the connection is not established and MD is Android device, MD will send MSG_CMD_START_BT_AUTOPAIR to inform HU to start auto pairing process **while MD shall start BT pairing related process**. Therefore, when to activate auto pairing process is determined by MD.

- When HU receives MSG_CMD_START_BT_AUTOPAIR command from MD, HU shall send its Bluetooth MAC address, PIN CODE (if PIN CODE mode is used) information to MD. After MD finishes a series of relevant initialization operations (such as automatically activating Bluetooth, checking Bluetooth connection status), it moves to the next step.
- MD shall send its Bluetooth MAC address, name (optional) and relevant information to HU. This could let HU know MD'S Bluetooth is in standby state (existing connection has been disconnected). Similarly HU has to check whether its Bluetooth is connected; if HU is not connected with this MD, HU shall disconnect the existing BT connection and standby for pairing request.
- HU shall check whether it has been connected to this MD before according to MD address and notify this information to MD so that MD can choose to start pairing or start connecting request.
- Then MD shall start pairing or start connecting request; after the pairing or connection is successful, MD shall notify MD of the connection status.

For details, please refer to the following figure:

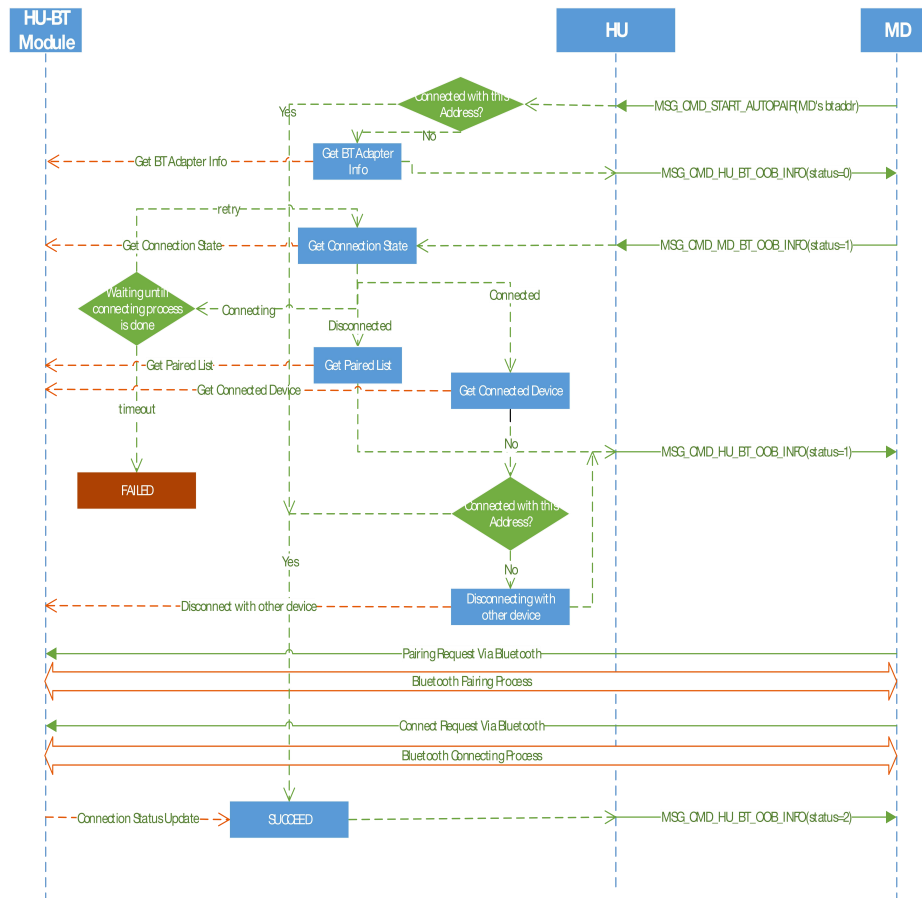


Figure 6.2-2

6.2.3. Auto-pair Initialization

Before implementing BT auto-pair and BT phone call functions, related initialization functions shall be completed:

- Inform features which MD supports (via MSG_CMD_HU_FEATURE_CONFIG_RESPONSE)
- Make sure whether CarLife BT phone call (with Internal UI) could be supported. HU sends START_IDENTIFY_REQ, and after MD receives it, it will reply IDENTIFY_RESULT_IND to inform HU whether it can start BT phone call functions. Currently, for Android phone with OS version prior to 5.0, MD does not support BT phone call with internal UI.
- After MD sends IDENTIFY_RESULT_IND message to HU, HU needs to block native BT phone call function, and waive the right to CarLife for implementing phone call.
- Before implementing BT phone call with internal UI, it needs to ensure HU connects to only one Bluetooth device (Certain HU supports to connect with multiple Bluetooth



devices. In this case, HU should disconnect with other devices before sending START_IDENTIFY_REQ)

The detailed process is listed in the following figure:

1. After CarLife USB connection has been established, the handshaking process between MD and HU will be activated. During this process, HU shall send its BT address to MD via message MSG_CMD_HU_INFO. Then MD shall send its BT address to HU via MD_INFO message (iOS will not send its BT address because it cannot obtain its BT address due to the system restriction).
2. In order to mark whether the connected BT device is CarLifedevic, HU is required to send START_IDENTIFY_REQ.
3. After MD receives START_IDENTIFY_REQ, MD will verify whether the HFP connection is successful with this HU. If the connection is established, then BT auto pairing will not be activated (for Android), and MD will send IDENTIFY_RESULT_IND to inform HU that the marking is successful and CarLife BT phone service can be activated.
4. If the HFP connection is not established and the MD is iOS, MD will prompt users to pair HU manually; for Android system, MD will inform HU to start the BT auto-pair process.
5. When HU receives the BT auto-pair command, it will disconnect existing HFP connection, and then start the BT auto-pair process.
6. MD will monitor the status of HFP connection with HU. If the connection is disconnected, MD will send IDENTIFY_RESULT_IND to inform HU, and end CarLife BT phone service.

(Notes: short messages are used above. MSG_CMD_BT_START_IDENTIFY_REQ = START_IDENTIFY_REQ
MSG_CMD_BT_IDENTIFY_RESULT_IN = IDENTIFY_RESULT_IND)

For details please refer to the following figure:



6.2.4. Auto-pair Interface Definition

6.2.4.1. MSG CMD HU BT OOB INFO

Field Name	Type	Optional/ Required	Description
address	String	Required	Bluetooth address of HU in 48 bit.
passkey	String	Optional	With PASSKEY comparison mode, this filed doesn't require input. With PIN CODE mode, required PIN Code should be input.
hash	String	Optional	A field reserved for OOB mode
randomizer	String	Optional	A field reserved for OOB mode
uuid	String	Required	Unused filed
name	String	Required	Bluetooth OF HU's name



Status	Integer	Required	This indicates the various status during BT auto-pair: IDLE = 0: Initiated status READY = 1: Means Bluetooth adapter has already disconnect existing connection and is ready for responding to new connection request CONNECTED = 2: Means HU has been connected with target MD over Bluetooth
ispaired	Integer	Optional	Used for indicating whether target MD has ever connected with HU before.

Table 6.2-1

6.2.4.2. MSG_CMD_MD_BT_OOB_INFO

This message will be sent to HU from MD to inform HU of MD's Bluetooth address and status information. For definition details, please refer to the following table.

Field Name	Type	Optional/ Required	Description
btAddress	String	Required	MD's Bluetooth address
passKey	String	Optional	Unused filed
hash	String	Optional	A field save for OOB mode
randomizer	String	Optional	A field save for OOB mode
uuid	String	Optional	Unused filed
name	String	Optional	Bluetooth name of MD
status	Integer	Required	Indicate status of Bluetooth auto-pair process. For details, please refer to Table 8.2-1.

Table 6.2-2

6.2.4.3. MSG_CMD_START_BT_AUTOPAIR

This message will be sent to HU from MD to inform HU of starting BT auto-pair process and disconnecting existing HFP connection. iOS MD will not send this message. For definition details, please refer to the following table.

Field Name	Type	Optional/ Required	Description
ostype	Integer	Required	Mobile Device OS:



			0: Android 1: iOS (Not Used)
address	String	Optional	Only applicable to Android OS, MD BT address shall be written.

Table 6.2-3

7. Video Engine

7.1. Processing of bdsc

Bdsc support is unnecessary for Android 5.0 or beyond but applicable to system below Android 5.0. After Android OS MD is connected with HU, HU needs to push an executable file (bdsc) provided by Baidu to MD, and run it on the MD to activate CarLife video-related function.

Based on different versions of Android systems on MD, different versions of bdsc are applied to match them. The table below displays the matchup between bdsc and different versions of Android system.

Android System Version of MD	bdsc file
Android 4.1	bdsc16
Android 4.2	bdsc17
Android 4.3	bdsc18
Android 4.4.2	bdsc19
Android 4.4.4	bdsc19_01

When connection is set up, HU applies adb command to push corresponding bdscXX to MD, and get operation flow of messages like the system version of Android MD. For details, please refer to Chapter Connection Engine. Related bdsc commands are shown as follows:

```
adb push bdscXX /data/local/tmp/bdsc
```

To ensure there is no remaining bdsc process, check the process list of MD first and then kill bdsc process:

```
adb shell ps
adb shell kill -9 (residual bdsc process number)
```

Then apply adb command to run it

```
adb shell chmod 777 /data/local/tmp/bdsc
```



```
adb shell /data/local/tmp/bdsc
```

After bdsc is run successfully, it will be able to send the message of starting video transmission, and to complete the subsequent connection process.

7.2. Dynamic FPS Mechanism

CarLife dynamic FPS (Frames Per Second) mechanism consists of dynamic FPS adjustment on HU and dynamic FPS adjustment on MD.

7.2.1. FPS Adjustment on HU

For HU whose decoding capability is less than 30 fps, or whose decoding capability decreases after running for a whole, CarLife communication protocol offers FPS adjustment protocol to manage the maximum speed of frame rate on MD.

The parameters of FPS adjustment sent by HU are only valid from 3 to 30.

For details, please refer to the **Communication Protocol** **MSG_CMD_VIDEO_ENCODER_FRAME_RATE_CHANGE**: related messages

7.2.2. FPS Adjustment on MD

The maximum FPS on MD cannot exceed the FPS adjustment parameter sent from HU. FPS on MD shall range in a reasonable scope according to the FPS adjustment strategy on MD.

7.3. Dynamic Resolution Mechanism

CarLife dynamic resolution mechanism is set up for adapting to different types of MD.

After MD has established communication with HU, HU will send the message of video resolution parameter that HU intends to get to MD. Once MD receives those messages, MD will figure out the video resolution parameter it could support, and give feedback to HU. **Due to the limitation of MD encoding module conditions, it cannot completely match HU's width parameter or its own aspect ratio of display area, the final parameters will be a set of width/height parameter pair as close to HU's requirement as possible.** Details of the message sequence can be seen in [Video Message Sequence When Setting Up Connection](#).

After MD interacts with HU on the message mentioned above, HU can apply the video parameters sent from MD, decoding and displaying the video data in video channel.



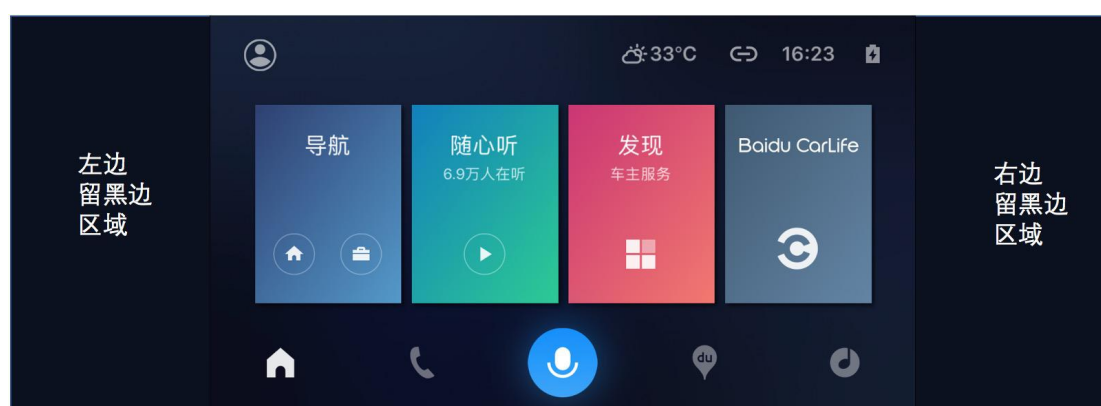
If the final parameters sent from MD are different from those sent from HU, HU shall reasonably zoom and stretch the displayed images according to its own resolution strategy.

7.4. Adaptation Strategy for Extra-wide or Extra-high Screens

When HU initially sets up connection with MD, they will decide the video resolution according to CarLife Protocol Field. CarLife app supports screen adaptation of standard screen (aspect ratio 16:9) and wide screen (aspect ratio 8:3). The MD default resolution is 16:9. If the HU is wide screen (8:3) and expects CarLife to adapt its wide screen with full screen display, OEM shall inform Baidu when applying for project channel number. Baidu will configure the project ID with wide screen in Baidu server, then the HU of this project ID will support CarLife wide screen with full screen display.

For extra-wide or extra-high screens whose aspect ratio is quite different from that of MD screen, full-screen display will result in over-stretch or over-compression. It is suggested that HU shall set a reasonable display area according to extra-wide (or extra-high) screens so that blank strips can be left on left & right (or upper & lower) sides, or display specific information. Then HU shall transmit the width/height parameters of display area to MD; then HU shall compress video images according to suitable compression strategy to fill HU's assigned display area.

For extra-wide screens, upper & lower sides shall be filled while symmetric blank strips shall be left on left & right sides.



Similarly, for extra-high screens, left & right sides shall be filled while symmetric blank strips shall be left on upper & lower sides.



If the blank strips area is too big, they can be used to display self-defined information of OEMs, such as HU native system main menu, or data, clock, etc.

7.5 Suggestion for HU Decoding & Display

Due to system coding characteristic of some Android phones, the image of the first I frame is black screen (or white screen) when the phone is encoding video, so the MD CarLife app cannot identify it. It is suggested that the video data between the first and the second I frames (or the first 500 milliseconds) is decoded but not displayed after MD and HU are connected.

8. Audio Engine

8.1. CarLife Audio Engine Architecture

Audio modules in CarLife mainly include Navi TTS Module, Media Module and VR Modules as shown in Figure 8.1-1:

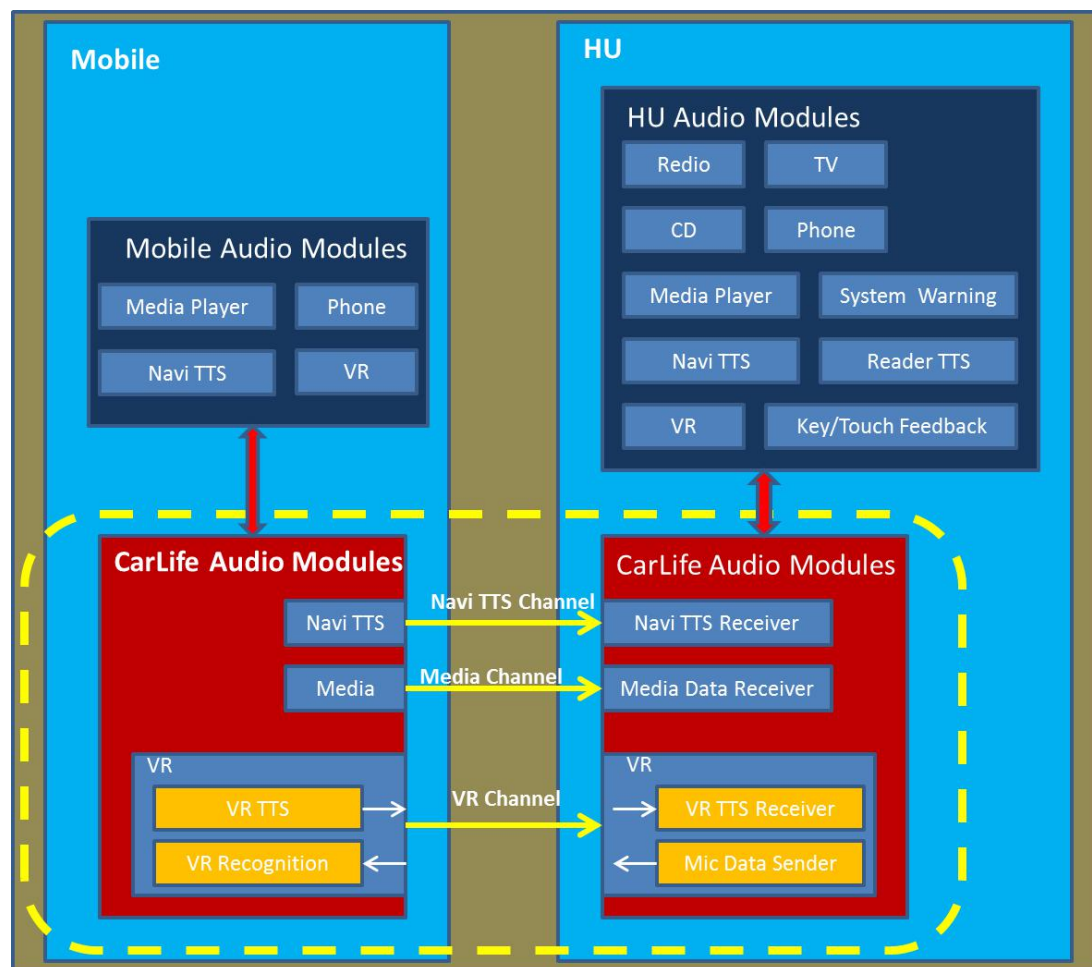


Figure 8.1-1

Among which,

Navi TTS Module can transmit relevant audio data stream (in PCM format) to HU via Navi TTS channel for corresponding HU modules to play.

Media audio Module can transmit relevant audio data stream (in PCM format) to HU via Media channel for corresponding HU modules to play.

VR Module can transmit VR prompt to HU via VR channel for corresponding HU modules to play; at the same time, VR module will receive microphone recorded data from HU via VR channel for voice recognition.

8.2. Audio Stream Types

CarLife has 4 types of audio streams, which are transmitted via 3 independent channels (VR TTS and VR MIC use the same channel at different direction). Detailed parameters and description are listed in Table 8.2-1:



Audio Streams	Format	Channel	Description
Media	Encoding format: PCM; Sampling rate: depends on which media data is going to be played on MD (normally 44.1kHz); Number of Channels: depends on the media data to be played on MD (normally dual channels); Quantization format: 16bits (little endian)	Media Channel MD to HU	MD will decode the compressed audio data (such as MP3, aac) to PCM data stream and transmit it to HU to play.
Navi TTS	Encoding format: PCM; Sampling rate: 16kHz; Number of Channels: mono; Quantization format: 16bits (little endian)	Navi TTS channel MD to HU	MD sends Navi TTS prompt to HU to play
VR TTS	Encoding format: PCM; Sampling rate: 16kHz; Number of Channels: mono; Quantization format: 16bits (little endian)	VR channel MD to HU	MD sends VR prompt to HU to play.
VR MIC	Encoding format: PCM;	VR channel	HU sends MIC recorded audio data



	Sampling rate: 16kHz; Number of Channels: mono; Quantization format: 16bits (little endian)	HU to MD	to MD, for VR engine to parse
--	--	----------	----------------------------------

Table 8.2-1

8.3. Audio Focus Types

Audio focus means the exclusivity on speaker gained by audio module (like music and TTS prompt). Before playing audio data, audio module needs to send request for audio focus. If audio focus is gained successfully, the audio module can play audio via the speaker. If not, it means audio can NOT be played.

CarLife defined 3 types of audio focus:

- Unknown exclusive
- Transient exclusive
- Mixable

Among which,

Unknown exclusive:

When CarLife sends request for this type of audio focus, CarLife will give an indication of request for audio play to the system for long duration. This type of audio focus is mainly used for media play, such as CarLife 'local music', 'Lebo', 'Netease Music' and third party audio SDK services like 'Ximalaya FM'. After this type of audio focus is gained, it will not be released automatically. It will be released only when other audio services take over the audio focus.

Transient exclusive:

When CarLife sends request for this type of audio focus, CarLife will send info to the system, indicating a temporary request for audio play device. When audio services play finishes, audio focus will be released automatically, and inform system that the temporary use of audio play device ends. This type of audio focus is mainly used in CarLife VR. When VR starts, this type of audio focus is applied for; when VR ends, this type of audio focus will be released.

Mixable

When CarLife sends request for this type of audio focus, CarLife will send notification to the system, indicating a transient request for audio play device. But when the audio play device is occupied, other audio services can be played at the same time, but the volume of other audio services should be lowered to ensure the mixable audio play have higher priority when multiple



audio services playing. This type of audio focus is mostly applied in Navi TTS broadcast. When TTS prompt occurs, Navi TTS prompt should have higher priority and run at foreground, and other audio services should reduce volume. When tts prompt ends, audio focus will be released.

Relation between CarLife Audio Services and Audio Focus is shown as below:

Audio Module	Audio Focus type
Media	Unknown Exclusive
Navi TTS	Mixable
VR	Transient Exclusive

Table 8.3-1

8.4. Audio Focus Management Strategy

When CarLife audio modules co-exist with other audio services, it should follow the following audio focus strategies:

1. Media Audio Module

- 1) Audio Focus Request:** 'Media' audio module (like "Local Music") in CarLife should send request for unknown exclusive audio focus before starting to play. When request fails (like phone call is ongoing), 'Media' audio module should pause.
- 2) Audio Focus Release:** 'Media' audio module will NOT automatically release audio focus. Only when other types of audio services take over audio focus (like other music player is playing), the audio focus will be released.
- 3) Audio Focus Monitoring:** When it has been monitored that audio focus of CarLife 'Media' audio module has been taken over by ' unknown exclusive audio services ' (other music player) or 'transient exclusive audio focus' (VR), 'Media' audio module will pause; when this audio module regains the audio focus (VR action ends), 'Media 'will continue to play. When it has been monitored that 'mixable' audio module (Navi TTS) takes over audio focus, 'Media' audio module should reduce volume, and start audio mixing with the audio module trying to occupy the audio focus.

2. Navi TTS Audio Module

- 1) Audio Focus Request:** before Navi TTS prompt plays, it can send request for 'mixable' audio focus; if the request fails (such as phone call is ongoing), TTS prompt should stop;



- 2) **Audio Focus Release:** after navi TTS play ends, audio focus will be automatically released.
- 3) **Audio focus monitoring:** When it is monitored audio focus of 'Navi TTS' module 'is taken over by 'Transient Exclusive' audio service (such as activating VR), Navi TTS prompt will not play.

3. VR TTS Audio Module

- 1) **Audio Focus Request:** VR TTS sends request for 'unknown exclusive' audio focus before playing; when request fails (such as phone call is ongoing), VR TTS prompt should stop.
- 2) **Audio Focus Release:** When VR TTS prompt ends, audio focus should be released automatically.
- 3) **Audio Focus Monitoring:** When it is monitored the audio focus of VR TTS module is taken over by 'transient exclusive' audio services (such as incoming call), VR TTS should stop playing.

8.4.1. Duck Process

When CarLife Media module and Navi are working at the same time, Media module's audio data and Navi TTS prompt will be transmitted to HU via Media channel and Navi TTS channel. HU will process Media PCM data and Navi TTS PCM data using its own audio mixing strategy and play them. In this case, 'Media' should reduce volume to make sure drivers can hear the Navi TTS. When Navi TTS and Media audio are played at the same time, this process can be referred to the following figure:

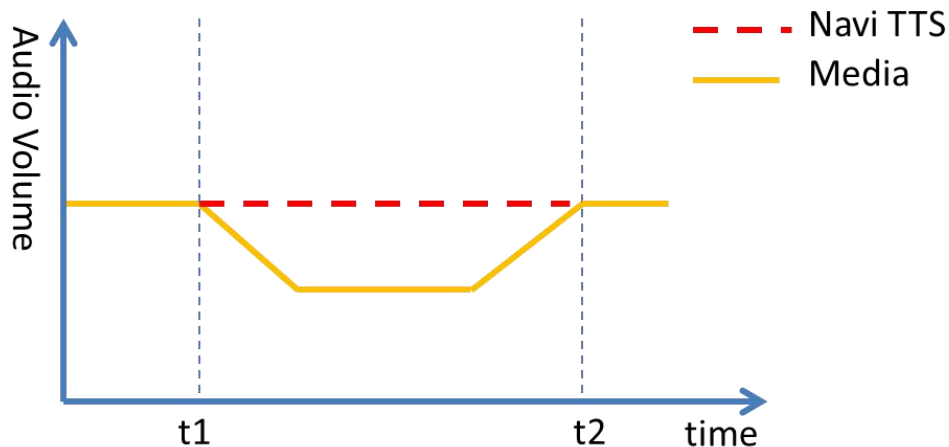


Figure 8.4-1: Strategy in Simultaneous Play of Navi TTS and Media

In this figure, NAVI TTS starts to play at t1, and stops at t2.



8.4.2. Mute Strategy

During the running of CarLife on HU, when HU's high priority audio services want to fully take over the speaker, it is necessary to stop audio playing in CarLife. There are two ways to implement it:

1. When HU receives Navi TTS, VR TTS, Media data from MD, HU can directly discard the data and choose NOT to play. In this case, MD will keep sending audio data to HU.
2. HU can use certain command via command channel MSG_CMD_MODULE_CONTROL, to stop CarLife' audio module (such as Navi, VR, and Music). In this case, music module can be resumed via this command apart from pause. When CarLife audio pause, module status change can be transmit to HU via command MSG_CMD_MODULE_CONTROL.
3. Mute mechanism during the phone call refers to users mute HU local mic during the phone call so that phone far-end cannot hear the voice of near-end; it includes 2 situations:

1) HU has integrated CarLife BT Phone function:

During the whole phone call, users can operate on built-in phone interface provided by CarLife. On one hand, users can click the Mute button on the interface to send mute mic command defined in BT phone (please refer to Section 11.3.2.1) via USB channel and disable local MIC.

On the other hand, users can press Hard Key on HU to disable local MIC. In this case, HU shall send corresponding KEY CODE to MD when users press the Hard Key so that MD can update its display on the interface.

2) HU uses Native BT Phone APP:

In this case, users use HU Native phone interface during the whole phone call. Users can click the Mute button on the interface or press HU Hard Key to mute MIC. The status shall be handed by HU itself.

8.5. Strategy in Conflict of HU Native Audio and CarLife Audio

When CarLife audio services on HU is running with the native audio services at the same time, HU audio services should conform to the following strategies:

1. Send request for audio focus before audio playing.
2. After audio play ends, it should decide whether to release audio focus according to what types of audio modules is taking over the audio focus.



- ✧ For 'Unknown Exclusive' audio focus (such as music play), it will not release audio focus automatically;
 - ✧ For 'mixable' audio focus (such as Navi TTS prompt), when ending exclusivity, it should release audio focus immediately;
 - ✧ For 'Transient Exclusive' audio focus (such as CarLife Voice Recognition), when ending exclusivity, it should release audio focus immediately.
3. When audio services are playing, HU should monitor the audio focus of CarLife audio module and respond accordingly.

8.6. Audio Services Priority Instruction

Audio focus provides status synchronization mechanism for different audio services in the system, which, however, cannot satisfy all audio scenarios. Therefore, apart from audio focus mechanism, priority for different audio modules shall also be considered. In CarLife audio system, audio modules priority is shown as below:

Phone > VR > Media/Navi

Phone module has the first priority. Media and Navi TTS prompt have the lowest priority (FM, BT Music and Local Music are all Media audio modules). When phone module is working, other audio services can't take over audio focus. When VR module is starting, Media and Navi TTS audio services can't take over audio focus.

8.7. Voice Recognition

CarLife Voice Recognition (CarLife VR) is developed based on SDK offered by Baidu Voice Recognition Department. Basic functions of CarLife VR include voice recognition and voice wake up. Voice recognition integrates online & offline recognition and integration, and voice wake up uses offline recognition.

Currently, only Mandarin voice commands are supported.

CarLife has following requirements for input voice materials:

- 1) Sampling rate: 16k
- 2) 16bit depth (little endian, signed magnitude)
- 3) Mono channel
- 4) Amplitude: about 5000
- 5) Signal-to-noise ratio: above 12db
- 6) Loudness Peak: no more than 0dBFS RMS



According to requirements mentioned above, partners should assess audio material recorded by HU microphone. For details, please refer to *Instruction to Assessing CarLife Record Materials*.

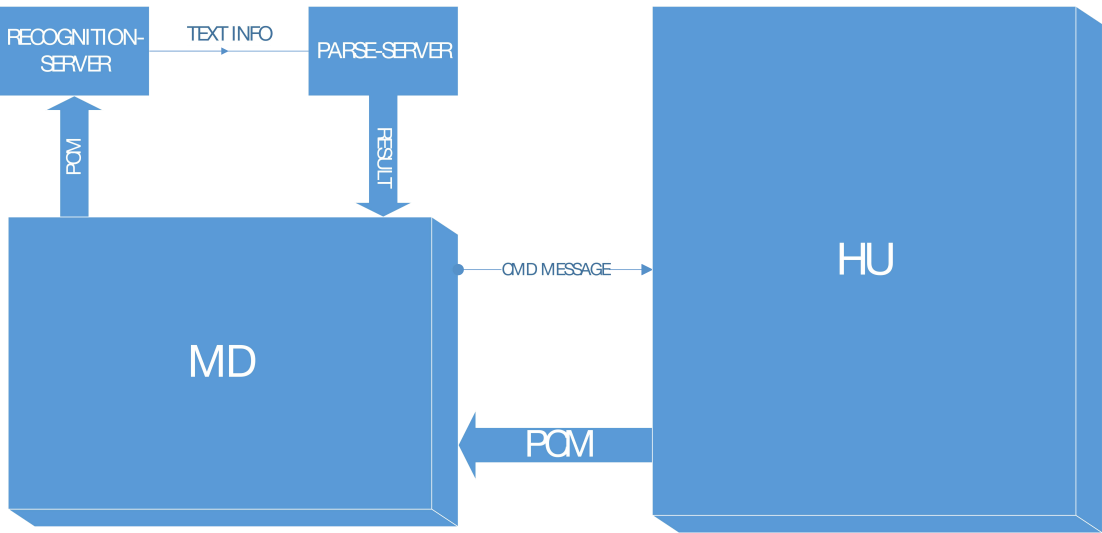


Figure 8.7-1: VR Architecture

When CarLife is not connected to HU, CarLife-VR will use MD's microphone to record input voice. When CarLife is connected, CarLife VR will use HU mic to record voice input by default. Voice input will be transmitted in PCM format to VR server, waiting voice server to return the result. For details, please refer to 'Chapter 12.6 Voice Message'

If HU mic can't meet CarLife VR requirements, CarLife allows OEMs to use MD's mic for recording. For details, please refer to 'Chapter 12.6 Voice Message'.

When CarLife launches for the first time, voice wake up shall be off by default. Users can turn on/off the switch of voice wake up in settings of CarLife app.

CarLife voice commands will continue to increase, improve and optimize. This will only rely on MD. Please refer to the voice commands supported by the latest MD.

The following are some commands that CarLife supports (currently only Mandarin is supported):

前往 Go to Certain Modules	前往 Go to home page
	前往电话 Go to phone
	前往地图 Go to map
	前往音乐 Go to music
音乐 Music	播放音乐 Play music
	上一首 Last song
	下一首 Next song



		播放 Play	
		我要听 XXX (人名)的歌 I want XXX (singer)'s song	
		我要听 XXX (歌名) I want XXX (song name)	
	地图 Map	回家 Go home	
		去公司/回公司 Go to company	
		附近的 XXX (加油站等) Where is nearby gas station	
		导航去 XXX Go to 'certain place'	
		我要去 XXX I want to go to 'certain place'	
	电话 Phone	打电话给 XXX (人名)Call XXX (name)	
		打电话给 XXX (电话)Call XXX (phone number)	
	帮助指令 Help Guidance	帮助 (展开语音帮助页) Help (open voice help guidance page)	
		打开小度语音唤醒/关闭语音唤醒 Enable Baidu voice wake	
		关闭小度语音唤醒/关闭语音唤醒 Disable Baidu voice wake	
		检查更新 (安卓)Check update (for Android)	

9. Bluetooth Telephone

9.1. Foreword

To improve user experience with CarLife during making and receiving calls, due to Android and iOS system restriction, many phone-related events & actions can't be captured by CarLife APP, so it requires HU as Bluetooth Hand-free Device to achieve this goal.

Due to the policy change of user privacy rights in Android, Android (part 8.0, all 9.0 and above) system could not get the Bluetooth address of MD. As most of the functions in this chapter rely on the Bluetooth address to complete, so it is not necessary for the HU to support this chapter per Android MD. And for functions relying on the MD CarLife to provide Bluetooth address, HU should evade accordingly.

There exists a default Bluetooth app in common HU system, so it is needed to cooperate with the HU manufacturer to complete relevant functions.



The premise of enabling Bluetooth phone call function requires that Bluetooth HFP connection between HU and HU CarLife MD been successfully established, built-in Bluetooth phone call function will not be activated.

Currently, only Android system 5.0 or beyond supports CarLife built-in Bluetooth phone call function, while all versions of IOS systems support built-in BT phone call function.

CarLife built-in Bluetooth phone call function can be greatly improved, and the user has a good CarLife user experience during the phone call, so it is strongly recommended that the manufacturers support this function.

FAQ about accessing CarLife's built-in Bluetooth phone function are as follows:

- Both MD with IOS system and MD in Android AOA connection mode support built-in Bluetooth phone call function
- When built-in Bluetooth phone call function is used, HU can only connect one Bluetooth device after CarLife is connected, and it does not support to connect with multiple Bluetooth devices
- At present, CarLife built-in Bluetooth phone does not support Multiparty Call, and in case of Multiparty Call, it will be handled by car-mounted native Bluetooth phone.
- After HU receives MSG_CMD_BT_IDENTIFY_RESULT_IND, if the result is TRUE, it shows that you can start using built-in Bluetooth phone call function, and it is needed to shield native Bluetooth phone interface of HU; if the result is False, it shows that built-in Bluetooth phone call function is not enabled.

Before starting CarLife built-in Bluetooth phone call function, it is needed to complete the initialization; for details, refer to Section 6.2.3

9.2. Interface Structure

9.2.1. Overall Structure

As the following figure shows, for iOS and Android systems, CarLife and Telephone don't have well-connected interfaces (for instance, Android device doesn't offer the APP to allow a third party to answer phone call). To implement Bluetooth telephone in CarLife, it requires sending certain messages to HU for HU to finish relevant operation.

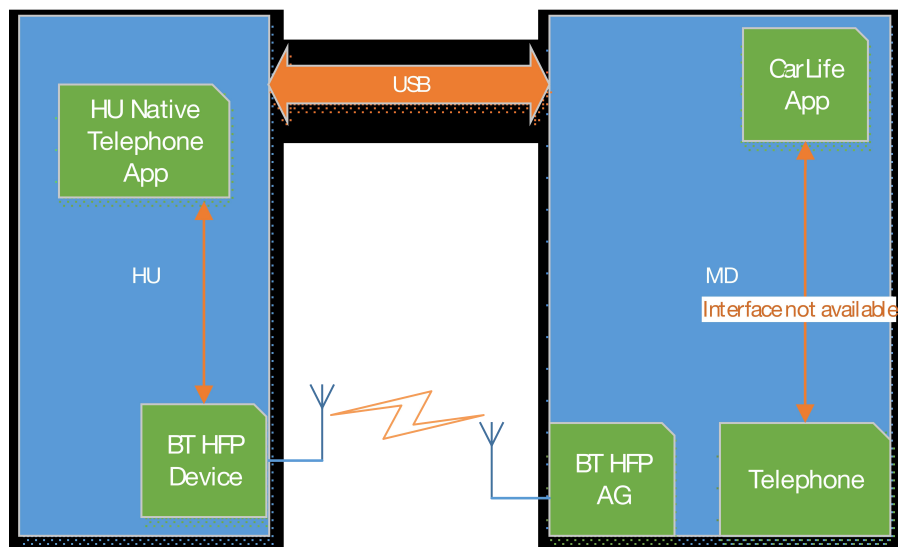


Figure 9.2-1

9.2.2. Bluetooth Telephone Message Sequence

The following figure presents the message flow of a classic scenario of picking up incoming call (due to android & iOS system restriction, CarLife APP can't pick up incoming call directly, so it requires HU Bluetooth as HFP client to pick up the call).

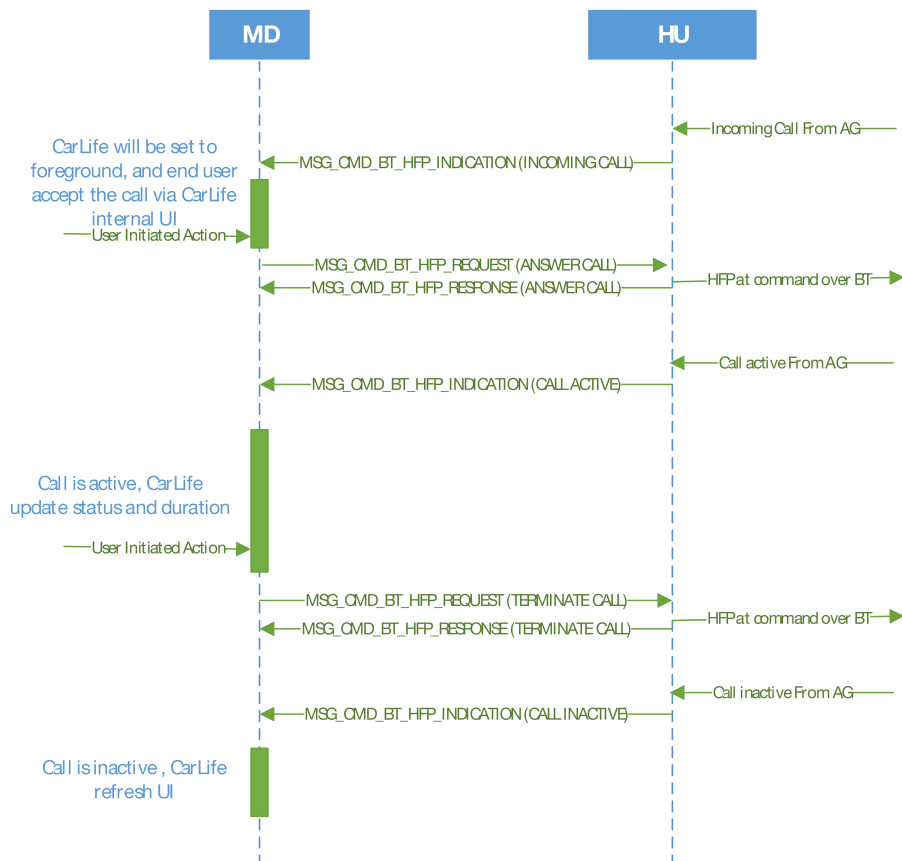


Figure 9.2-2

When HU and MD are connected via Bluetooth, a typical incoming call process should proceed through following steps:

- When HU's Bluetooth receives an incoming call indication, it should use pre-defined PB message to inform CarLife of the incoming call status.
- CarLife APP will replace the native Bluetooth telephone App, and pop up an incoming calls' tips in CarLife internal telephone UI.
- User can select 'answer' button or 'reject' button, and relevant PB message will send to HU. Once HU receives that message, it will send standard HFP AT Command to pick up or reject phone call.
- When phone has been picked up, HU should update the call status to MD.
- When phone call finished, user can click 'terminate' button to inform HU to send HFP standard hang up command.
- In the end of sequence, HU will update telephone status to CarLife on MD as well



9.3. Interface Definition

9.3.1 Telephone Functions

- Answer An Incoming Call
- Reject An Incoming Call
- Place A Call With Phone Number
- Terminate Call Process
- Transmit DTMF Codes
- Incoming call notification
- Place a call notification
- Call active or inactive notification
- Mute MIC
- After HU finishes handshaking with CarLife, if Bluetooth has been connected, HU can use CarLife Bluetooth telephone (known as telephone internal UI) and stop HU native Bluetooth APP to run in foreground. Necessary telephone event will be sent to CarLife on MD via USB/WIFI protocol.
- If CarLife on MD quits, HU should be able to resume native Bluetooth APP for related operating.
- Multi-party phone call functions are not supported currently. If an incoming call happens when a phone call is ongoing, native telephone app should popup to deal with this situation.
- Hard keys for telephone function are supported for some HU. HU should take the responsibility to send the key code to MD when key is pressed, however associated key function need to be handled by HU itself.
- For phone service related HU hardkey, when users press the hardkey, corresponding keycode should be sent to CarLife on MD. The execution should be done by HU.

9.3.2. Detailed Interface Definitions

9.3.2.1. MSG_CMD_BT_HFP_REQUEST

This message will be sent to HU from MD, requesting HU to execute phone call related commands via its own Bluetooth HFP protocols. For details, please refer to the following table:

Field Name	Type	Optional/ Required	Description
command	Integer	Required	Clarify which telephone command



			is going to be sent
phonenum	String	Optional	When making a phone, this field should clarify the telephone number
dtmfcode	Integer	Optional	Clarify DTMF code to be sent

Table 9.3-1

Main commands include:

Name	Value	Usage
PLACE_OUTGOING_CALL	1	make a phone call
TERMINATE_CALL	2	hang up the phone
ANSWER_CALL	3	pick up the phone call
REJECT_CALL	4	refuse incoming call
SEND_DTMF	5	send DTMF code
MUTE_MIC	6	mute local MIC
UNMUTE_MIC	7	Unmute local MIC

Table 9.3-2

9.3.2.2. MSG_CMD_BT_HFP_INDICATION

This message will be sent to MD from HU to inform MD of telephone status after Bluetooth connection. For details, please refer to the following table:

Field Name	Type	Optional/ Required	Description
State	Integer	Required	Clarify telephone status, for details, please refer to Table 11-4
Phonenum	String	Optional	When receiving incoming call, this field clarifies incoming call's number
Name	String	Optional	unused field
Address	String	Optional	When MD receives Phone status message, it requires HU to fill out this field in order to indicate the status source of this BT device.

Table 9.3-3

Main status includes:

Name	Value	Usage
INCOMING_CALL	1	indicate new incoming call
OUTGOING_CALL	2	indicate a successful outgoing call
CALL_ACTIVE	3	indicate phone call activated
CALL_INACTIVE	4	indicate phone call terminated
MULTICALL_ACTIVE	5	indicate multi-call activated



Table 9.3-4

9.3.2.3. MSG_CMD_BT_HFP_CONNECTION

When MD and HU have successfully set up USB connection, and Bluetooth HFP connection status changes, this message can inform MD of Bluetooth connection status.

For details, please refer to the following table:

Field Name	Type	Optional/ Required	Description
State	Integer	Required	When Bluetooth HFP connection status changes, this field indicates specific status of Bluetooth HFP connection. For details, please refer to Table 11-6.
Address	String	Optional	Address of connected or disconnected target device.
Name	String	Optional	unused field

Table 9.3-5

Main status definition includes:

Name	Value	Usage
HFP_DISCONNECTED	0	indicate HFP disconnected
HFP_CONNECTING	1	indicate HFP connecting
HFP_CONNECTED	2	indicate HFP connected

Table 9.3-6

9.3.2.4. MSG_CMD_BT_HFP_RESPONSE

This message will be sent to MD from HU to feedback and inform MD of the results of previous request. MD could update related interface status based on the results.

For details, please refer to the following table:

Field Name	Type	Optional/ Required	Description
Status	Integer	Required	Indicating the results of specific command 0: Command fails 1: Command succeeds
cmd	Integer	Required	Indicating which command is executed, for details, please refer



			to the following table.
Dtmfcode	Integer	Optional	If DTMF code is sent successfully, HU will be required to give response regarding which DTMF code is sent, including number 0-9, *, #

Table 9.3-7

Main commands include:

Name	Value	Usage
PLACE_OUTGOING_CALL	1	Making a phone call
TERMINATE_CALL	2	Hang up calls
ANSWER_CALL	3	Answer calls
REJECT_CALL	4	Refuse calls
SEND_DTMF	5	Send DTMF code
MUTE_MIC	6	Mute local MIC
UNMUTE_MIC	7	Unmute local MIC

Table 9.3-8

9.3.2.5. MSG_CMD_BT_HFP_STATUS_REQUEST

This command is sent to HU from MD to obtain specific status. Currently it is mainly used for obtaining whether HU MIC is mute or not.

Field Name	Type	Optional/ Required	Description
Type	Integer	Required	Obtaining specific status: 1: Obtaining MIC mute status

Table 9.3-9

9.3.2.6. MSG_CMD_BT_HFP_STATUS_RESPONSE

This command is sent to MD from HU to feedback specific status to MD when HU has received STATUS_REQUEST from MD. It is mainly used to feedback whether HU MIC is mute or not.

Field Name	Type	Optional/ Required	Description
Status	Integer	Required	Indicating the result of executing specific command. 0: MIC is unmute 1: MIC is mute
Type	Integer	Required	Indicating the type of current response. 1: MIC status

Table 9.3-10

9.3.2.7. MSG_CMD_BT_IDENTIFY_RESULT_IND

This command is sent to HU from MD to inform HU of the status of current HFP connection with HU BT. This message will only be sent when the HFP connection is connected or disconnected. When HU receives state SUCCESS, it needs to block the native Bluetooth phone call function, and CarLife built-in Bluetooth phone will take over the phone call function. When HU receives state FAILURE, it needs to activate the native Bluetooth phone to control the phone call function.

For details, please refer to the following table:

Field Name	Type	Optional/ Required	Description
state	Integer	Required	This field contains two status: SUCCESS and FAILURE. SUCCESS = 1: When HFP connection between MD and HU is established, MD will send this status to HU. FAILURE=0: When HFP connection between MD and HU is disconnected, MD will send this status to HU.
address	String	Required	The address of connected disconnected target devices, generally the BT address of HU.

Table 9.3-11

9.3.2.8. MSG_CMD_BT_IDENTIFY_RESULT_REQ

This message is sent to MD from HU to inform MD of whether the marked device is HU. Use message MSG_CMD_BT_IDENTIFY_RESULT_IND to inform MD of the status of marking.

For details, please refer to the following table:

Field Name	Type	Optional/ Required	Description
address	String	Required	BT address of marked HU

Table 9.3-12

9.4. Interaction Sequence of Critical Scenarios

9.4.1. Outgoing Call scenario

A typical outgoing call's sequence can be found in the following figure. It allows user to make a phone call via CarLife Internal Telephone UI. For details, please refer to description in following figure.

This scenario is only applicable to Internal Telephone UI of iPhone. If the connected MD is Android phone, **most of the MDs** will directly dial for outgoing call without informing HU via MSG_CMD_BT_HFP_REQUEST; HU **can** obtain call status via Bluetooth and inform MD via MSG_CMD_BT_HFP_INDICATION.

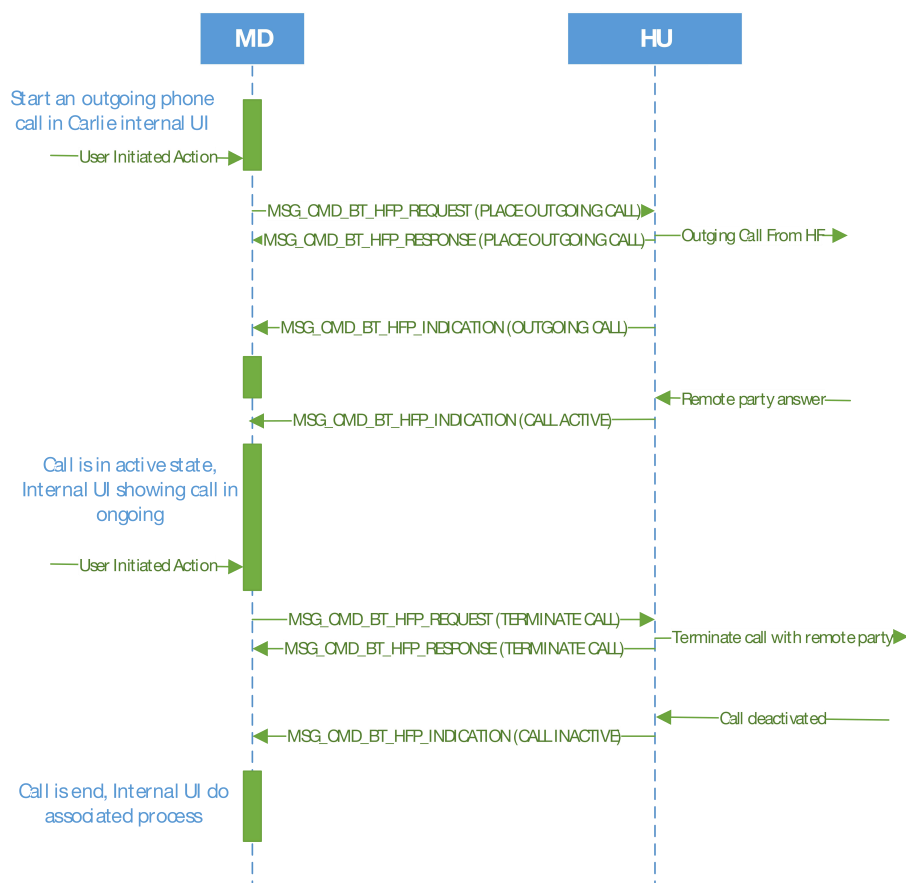
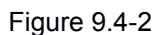


Figure 9.4-1

Sending DTMF during phone call can follow the sequence below. When users make a phone call via CarLife Internal Telephone UI, DTMF code will be sent. For detailed sequence, please refer to the following figure.



Once CarLife on MD successfully connects to HU via USB, and protocol versions match, Bluetooth HFP connection status should be sent to MD from HU. CarLife on MD will enable or disable CarLife Internal Telephone UI according to HFP connection status. If HFP connection is disconnected, Bluetooth telephone functions won't be activated.

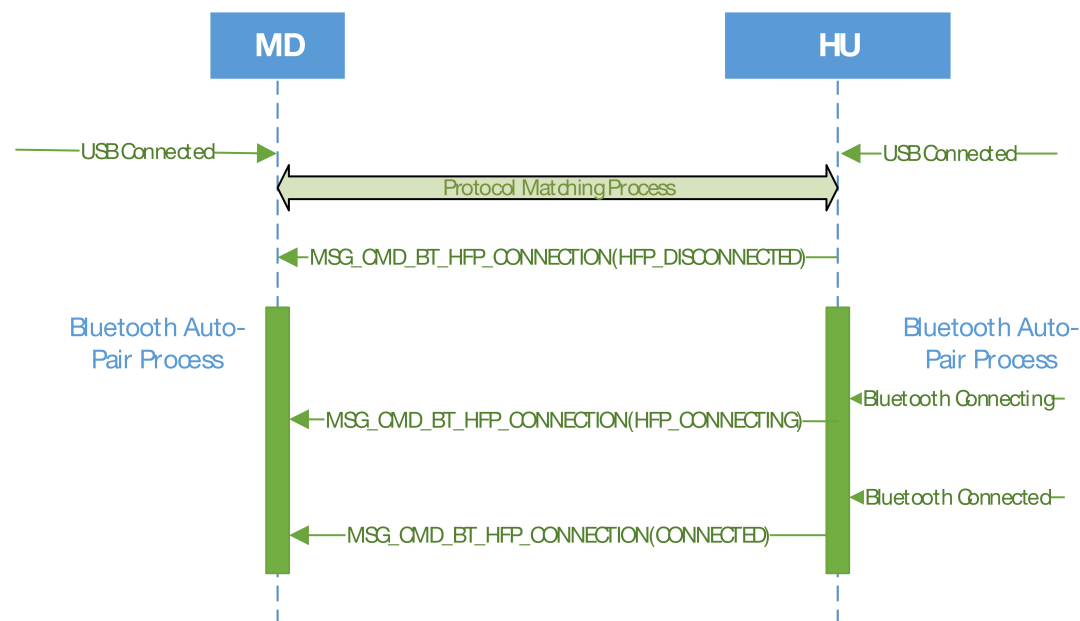


Figure 9.4-3

10. Communication Protocol

10.1. Naming Rules of Service Type

There is a service type field in the message structures of all channels to exclusively mark this message. Also, there are certain requirements on the command message of this field. The service type macro definition naming rule is: MSG_the name of channel_keywords of the function of service type. For example, the macro definition of Android system info sent from Android MD to HU is: MSG_CMD_MD_INFO.

The first two bytes of service type indicate different channels or major service ID, and the following two bytes indicate the value of service type. If the first figure of the next two bytes is 0, it means the message is sent from MD to HU; if the first figure of the next two bytes is 1, it means the message is sent from HU to MD. The ID of the first two bytes are shown in the following table:

Table10.1-1: ID of the First Two Bytes in Service Type of Channels

Channel Names	The First Two Bytes
Cmd	0x0001
Video	0x0002



Music	0x0003
TTS	0x0004
VR	0x0005
Touch	0x0006

10.2. Command Message

Command message in CarLife is transmitted via Cmd Channel, which mainly includes the synchronous message and simple data interaction among different modules.

10.2.1. Package Structure

CarLife command message structure is shown as below:

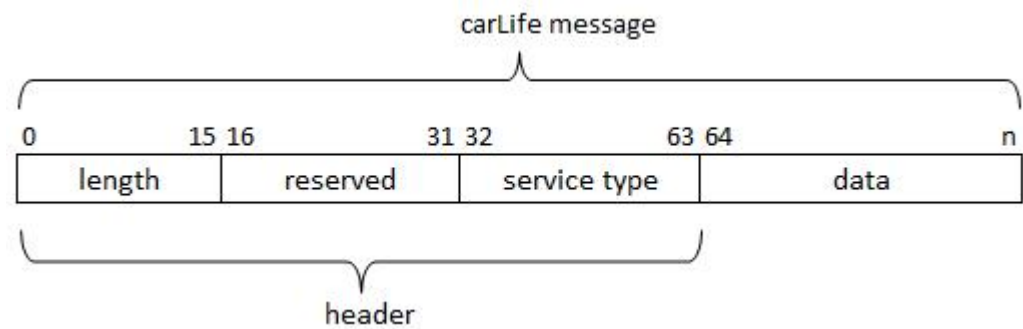


Figure 10.2-1: CarLife Command Message Structure

The length of CarLife command message header is fixed as 8 bytes. The definition of each field is shown in the following table:

Table 10.2-1: Field Definition of CarLife Command Message

Field Name	Offset	Length(bit)	Notes
length	0~15	16	The length of data field serialized by PB
reserved	16~31	16	Reserved field
service type	32~63	32	Message service type, For details, please refer to Related Messages.
data	64~n	length*8	The parameter of service type field. The byte stream serialized by PB.



10.2.2. Related Messages

The data fields which correspond to command message service type are all serialized by PB. The definitions of service type on HU and MD and its corresponding data field are shown as Table10.2-2:

Table10.2-2: Definition of CarLife Command Message Service Type and Data Field

Field Macro Definition Name	Service type	PB Structure of Data Field	Direction	Definition and Functions	Scenario
MSG_CMD_HU_PROTOCOL_VERSION	0x00018001	<pre> message CarlifProtocolVersion { required int32 majorVersion = 1; required int32 minorVersion = 2; } enum PBVersion { [majorVersion:1, minorVersion:0], [majorVersion:2, minorVersion:0] } </pre>	HU to MD	Once connected successfully, this command should be the first interactive command. HU sends its own protocol version info to MD. Then, MD will estimate the compatibility according the version number of MD. Version Info includes: majorVersion: the number of major version minorVersion: the number of minor version For all PBVersion versions, see enum definition, and the latest versions are [majorVersion: 2, minorVersion: 0].	The first message sent from HU to MD when connection is set up.
MSG_CMD_PROTOCOL_VERSION_MATCH_STATUS	0x00010002	<pre> message CarlifProtocolVersionMatch Status { required int32 matchStatus = 1; } </pre>	MD to HU	MD estimates the compatibility, matching status is shown as below: VERSION_MATCH = 1; VERSION_NOT_MATCH = 2;	The first message sent from MD to HU, when receiving MSG_CMD_HU_PROTOCOL_VERSION



MSG_C MD_HU _INFO	0x0001 8003	message CarlifeDeviceInfo { optional string os = 1; optional string manufacturer = 13; optional string model = 14; optional string release = 19; required string btaddress = 23; required string btname = 24; }	HU to MD	The HU system info sent from HU to MD. The OS field means the type of HU operating system, which could be QNX, Linux, Wince, Android	When connection is set up successfully, HU actively sends related info to MD.
MSG_C MD_MD _INFO	0x0001 0004	message CarlifeDeviceInfo { optional string os = 1; optional string board = 2; optional string bootloader = 3; optional string brand = 4; optional string cpu_abi = 5; optional string cpu_abi2 = 6; optional string device = 7; optional string display = 8; optional string fingerprint = 9; optional string hardware = 10; optional string host = 11; optional string cid = 12; optional string manufacturer = 13; optional string model = 14; optional string product = 15; optional string serial = 16; optional string codename = 17; optional string incremental = 18; optional string release = 19; optional string sdk = 20; optional int32 sdk_int = 21; optional string token = 22; optional string btaddress = 23; optional string btname =	MD to HU	System info of MD sent from MD to HU. Here, os field means the type of MD operating system, which could be Android, or IOS. Among other things, release field describes the version number of CarLife MD, and the value is a character string prefixed with "CarLifePhoneVersion:"	When connection is set up successfully, MD actively sends related info to HU.



		24; }			
MSG_C MD_HU _BT_OO B_INFO	0x0001 8005	message CarlifeBTOOBInfo { required string address = 1; optional string passKey = 2; optional string hash = 3; optional string randomizer = 4; required string uuid = 5; required string name = 6; required int32 status = 7; }	HU to MD	Bluetooth auto-pair info sent from HU to MD.	When handshaking process is successfully completed, HU sends this message to start Bluetooth auto-pair. For details, please refer to Chapter Bluetooth Auto-pair.
MSG_C MD_MD _BT_OO B_INFO	0x0001 0006	message CarlifeBTOOBInfo { required string btAddress = 1; optional string passKey = 2; optional string hash = 3; optional string randomizer = 4; required string uuid = 5; required string name = 6; required int32 status = 7; }	MD to HU	Bluetooth auto-pair info sent from MD to HU.	When MD receives the BT auto-pair info sent from HU, MD needs to respond this BT info via this message. For details, please refer to Chapter Bluetooth Auto-pair.
MSG_C MD_VID EO_EN CODER _INIT	0x0001 8007	message CarlifeVideoEncoderInfo { required int32 width = 1; required int32 height = 2; required int32 frameRate = 3; }	HU to MD	Request for video encoder initiation sent from HU to MD.	For details, please refer to Chapter Video Message.
MSG_C MD_VID EO_EN CODER _INIT_D ONE	0x0001 0008	message CarLifeVideoEncoderInfo { required int32 width = 1; required int32 height = 2; required int32 frameRate = 3; }	MD to HU	Message sent from Mobile Device to Head Unit that indicates the initiation of video encoder has successfully been done and indicates the resolution of the video.	For details, please refer to Chapter Video Message.
MSG_C MD_VID	0x0001	N/A	HU to MD	Message sent from HU to MD indicating	For details, please refer to



EO_EN CODER _START	8009			video transmission starts	Chapter Video Message .
MSG_C MD_VID EO_EN CODER _PAUSE	0x0001 800A	N/A	HU to MD	Message sent from HU to MD indicating video encoder pauses. MD deals with this message in the same way as it switches to the background.	For details, please refer to Chapter Video Message .
MSG_C MD_VID EO_EN CODER _RESET	0x0001 800B	N/A	HU to MD	Message sent from HU to MD that indicating the reinitiation of video transmission.	For details, please refer to Chapter Video Message .
MSG_C MD_VID EO_EN CODER _FRAM E_RATE _CHAN GE	0x0001 800C	message CarlifeVideoFrameRate { required int32 frameRate = 1; }	HU to MD	Message sent from HU to MD requesting to change the video frame rate.	For details, please refer to Chapter Video Message .
MSG_C MD_VID EO_EN CODER _FRAM E_RATE _CHAN GE_DO NE	0x0001 000D	message CarlifeVideoFrameRate { required int32 frameRate = 1; }	MD to HU	Message sent from MD to HU indicating that the change of video frame rate has successfully done, and returning the new frame rate info.	For details, please refer to Chapter Video Message .
MSG_C MD_PA USE_M EDIA	0x0001 800E	N/A	HU to MD	Message sent from HU to MD to pause Media play.	Abandoned
MSG_C MD_CA R_VELO CITY	0x0001 800F	message CarLifeCarSpeed { required uint32 speed = 1; optional uint64 timeStamp = 2; }	HU to MD	Message sent from HU to MD indicating current car velocity.	For details, please refer to Chapter Use of HU Data and Suggestions
MSG_C MD_CA R_GPS	0x0001 8010	message CarLifeCarGps { required uint32 antennaState = 1; // dBHz required int32 latitude = 3; }	HU to MD	Message sent from HU to MD indicating the current location. Here, WGS84 and GCJ02 coordinate system is	For details, please refer to Chapter Use of HU Data and Suggestions



		<pre>// 1/1,000,000 degrees required int32 longitude = 4; // 1/1,000,000 degrees required int32 height = 5; // 0.1 meters required uint32 speed = 6; // 0.01 km/h units required uint32 heading = 7; // 0.1 degrees // units 0.1 required uint32 pdop = 16; // units 0.1 required uint32 satsUsed = 18; required uint32 satsVisible = 19; optional uint64 timeStamp = 25; }</pre>		supported. The default value of sending frequency is 1Hz.	
MSG_C MD_CA R_GYR OSCOPE	0x0001 8011	<pre>enum GyroType { SINGLE_AXIS_GYRO = 1; THREE_AXIS_GYRO = 2; } message CarlifeGyroscope { required int32 gyroType = 1; required doublegyroX = 2; required doublegyroY = 3; required doublegyroZ = 4; optional uint64 timeStamp = 5; }</pre>	HU to MD	Gyroscope info sent from HU to MD When HU uses single-axis gyroscope, set gyroX as the value of single-axis gyroscope. The value of gyroY and gyroZ is set as 0. The value of gyroscope is angular velocity, and the Unit: degree/sec. gyroX: the angle between the projection on horizontal direction of Y axis and due north direction. The default value of sending frequency is 10Hz.	For details, please refer to Chapter Use of HU Data and Suggestions
MSG_C MD_CA	0x0001	<pre>message CarlifeAcceleration {</pre>	HU to MD	Car acceleration info sent from HU to	For details, please refer to



R_ACC ELERAT ION	8012	<pre> required doubleaccX = 1; required doubleaccY = 2; required doubleaccZ = 3; optional uint64 timeStamp = 4; } </pre>		MD. The unit of the accelerator value is m/s² accX: Accelerate speed in horizontal direction, positive to the right. accY: Accelerate speed in vertical direction, positive to the front. accZ: Accelerate speed in spatially vertical direction, positive to the sky direction. The default value of sending frequency is 10HZ.	Chapter Use of HU Data and Suggestions
MSG_C MD_CAR_OIL	0x0001 8013	<pre> message CarlifeOil { required int level = 1; required int range = 2; required bool lowFuleWarning = 3; } </pre>	HU to MD	The fuel consumption info sent from HU to MD	For details, please refer to Chapter Use of HU Data and Suggestions
MSG_C MD_SCREEN_ON	0x0001 0018	N/A	MD to HU	Message sent from MD to notify HU of turning on screen.	It is currently NOT used.
MSG_C MD_SCREEN_OFF	0x0001 0019	N/A	MD to HU	Message sent from MD to notify HU of turning off screen.	It is currently NOT used.
MSG_C MD_SCREEN_USERP RESENT	0x0001 001A	N/A	MD to HU	MD notifies HU that MD has been unlocked.	It is currently NOT used.
MSG_C MD_FOGREGRO	0x0001 001B	N/A	MD to HU	MD notifies HU that MD has been switched to	Please refer to



UND				foreground.	Chapter 11.1
MSG_C MD_BA CKGRO UND	0x0001 001C	N/A	MD to HU	MD notifies HU that MD has been switched to background.	Please refer to Chapter 11.1
MSG_C MD_LA UNCH_ MODE_ NORMA L	0x0001 801D	N/A	HU to MD	HU launches CarLife APP on MD and enter Home Page.	N/A
MSG_C MD_LA UNCH_ MODE_ PHONE	0x0001 801E	N/A	HU to MD	HU launches CarLife APP on MD and enters Phone Page.	N/A
MSG_C MD_LA UNCH_ MODE_ MAP	0x0001 801F	N/A	HU to MD	HU launches CarLife APP on MD and enters Map Page.	N/A
MSG_C MD_LA UNCH_ MODE_ MUSIC	0x0001 8020	N/A	HU to MD	HU launches CarLife APP on MD and enters Music Page.	N/A
MSG_C MD_GO _TO_DE SKTOP	0x0001 0021	N/A	MD to HU	Send this command when clicking the button “back to native system” on CarLife MD.	N/A
MSG_C MD_MIC _RECO RD_WA KEUP_S TART	0x0001 0022	N/A	MD to HU	MD notifies HU to activate voice wake-up, and start to record. (the sampling rate is 16K)	When voice wake-up is activated, this message is sent from MD.
MSG_C MD_MIC _RECO RD_EN D	0x0001 0023	N/A	MD to HU	MD notifies HU to pause recording.	When voice wake-up or voice recognition ends, this message is sent from MD.
MSG_C MD_MIC _RECO	0x0001 0024	N/A	MD to HU	MD notifies HU to start voice recognition and	When voice recognition launches, this



RD_RE COG_S TART				begin to record. (the sampling rate is 16K)	message is sent from MD.
MSG_C MD_GO _TO_FO REGRO UND	0x0001 8025	N/A	HU to MD	This message is sent from HU to MD to switch CarLife to foreground when launching CarLife	This message is used only when AOA connection is set up. In ADB connection, <u>ADB</u> <u>commands</u> will be used. In <u>iPhone</u> <u>Lightning</u> connection, relevant commands should be sent to underlayer. For details, please refer to relevant chapters.
MSG_C MD_MO DULE_S TATUS	0x0001 0026	enum ModuleId { CARLIFE_PHONE_MODUL E_ID=1; CARLIFE_NAVI_MODULE_I D=2; CARLIFE_MUSIC_MODULE _ID=3; CARLIFE_VR_MODULE_ID =4; //将 RECORD 关键字修改 为 VR CARLIFE_CONNECT_MOD ULE_ID=5; CARLIFE_MIC_MODULE_ID =6; CARLIFE_MEDIAPCM_MOD ULE_ID=7; CARLIFE_EDOG_MODULE _ID=8; (P/S: PB_VERSION 2.0) CARLIFE_CRUISE_ID=9; (P/S: PB_VERSION 2.0) } enum PhoneState	MD to HU	This message is used for synchronizing the status of MD and HU: MD notifies HU that the status of music module, VR module, and navi module has changed. Notes: VRState.VR_STAT US_NOT_SUPPOR T can be supported, but it is NOT recommended to apply. It is recommended to use MicState.MIC_STA TUS_NOT_SUPPO RT Enum ModuleId { CARLIFE_EDOG_ MODULE_ID=8; CARLIFE_CRUISE _ID=9;	Sent to HU when module status of MD changes. Module status can be referred to enum type description.



		<pre>{ PHONE_STATUS_IDLE=0; PHONE_STATUS_INCOMING=1; PHONE_STATUS_OUTING=2; PHONE_STATUS_CONNECTED=3; } enum NaviState { NAVI_STATUS_IDLE =0; NAVI_STATUS_START=1; } enum MusicState { MUSIC_STATUS_IDLE=0; MUSIC_STATUS_RUNNING=1; } enum VRState { VR_STATUS_RECORD_IDLE =0; VR_STATUS_RECORD_RUNNING =1; VR_STATUS_NOT_SUPPORT=2; // 有些车机不支持 VR } enum ConnectState { CONNECT_STATUS_ADB=1, CONNECT_STATUS_AOA=2, CONNECT_STATUS_NCM_ANDROID=3, CONNECT_STATUS_NCM_IOS =4, CONNECT_STATUS_WIFI=5, }</pre>		<pre>} These MODULE_ID are new added in PB_VERSION2.0</pre>	
--	--	--	--	--	--



		<pre> enum MicState { MIC_STATUS_USE_VEHICLE_MIC = 0; MIC_STATUS_USE_MOBILE_MIC = 1; MIC_STATUS_NOT_SUPPORTED = 2; } enum MediaPCMState { MEDIAPCM_STATUS_IDLE = 0; MEDIAPCM_STATUS_RUNNING = 1; } enum EDogState { EDOG_STATUS_IDLE = 0; EDOG_STATUS_RUNNING = 1; } enum CruiseState { CRUISE_STATUS_IDLE = 0; CRUISE_STATUS_RUNNING = 1; } message CarlifeModuleStatusList { required int32 cnt = 1; repeated com.baidu.carlife.protobuf.CarlifeModuleStatus moduleStatus = 2; } message CarlifeModuleStatus { </pre>			
--	--	--	--	--	--



		<pre> required int32 moduleID = 1; required int32 statusID = 2; } </pre>			
MSG_C MD_STATIS TIC_INFO	0x0001 8027	<pre> message CarlifeStatisticsInfo { required string cuid = 1; required string versionName = 2; required int32 versionCode = 3; required string channel = 4; required int32 connectCount = 5; requiredint32connectSu ccessCount = 6; required int32 connectTime = 7; optional string crashLog = 8; } </pre>	HU to MD	Statistic Info: cuid: the unique ID code of HU versionName: the version name of HU versionCode: the version code of CarLife on HU channel: the channel ID that assigned by Baidu to OEMs or HU manufacturers connectCount: connection times , which is dynamically saved by CarLife on HU. Once successfully connected, this parameter will be cleared connectSuccessCo unt: times of successful connection , which is usually set as 1 connectTime: Time for setting up connection crashLog: the key log of last crash on HU.	This message must be sent from HU to MD when connection is successfully set up. And cuid of each HU must be different. Otherwise, it will lead to auto disconnection. Each field must be filled correctly and be sent to MD. For details, please refer to Chapter Statistic Information.
MSG_C MD_MODULE_C ONTROL	0x0001 8028	<pre> enum ModuleId { CARLIFE_PHONE_MODUL E_ID=1; CARLIFE_NAVI_MODULE_I </pre>	HU to MD	This message is sent from HU to control different modules (music, VR, navi) on MD. Notes:	This message is sent to MD when HU needs to control the function of different



		D=2; CARLIFE_MUSIC_MODULE_ID=3; CARLIFE_VR_MODULE_ID=4; //将 RECORD 关键字修改为 VR CARLIFE_CONNECT_MODULE_ID=5; CARLIFE_MIC_MODULE_ID=6; CARLIFE_MEDIAPCM_MODULE_ID=7; CARLIFE_EDOG_MODULE_ID=8; (P/S: PB_VERSION 2.0 新增) CARLIFE_CRUISE_ID=9; (P/S: PB_VERSION 2.0 新增) } enum PhoneState { PHONE_STATUS_IDLE=0; PHONE_STATUS_INCOMING=1; PHONE_STATUS_OUTING=2; } enum NaviState { NAVI_STATUS_IDLE=0; NAVI_STATUS_START=1; } enum MusicState { MUSIC_STATUS_IDLE=0; MUSIC_STATUS_RUNNING=1; } enum VRState { VR_STATUS_RECORD_IDLE=0; VR_STATUS_RECORD		1. CARLIFE_MEDIAPCM_MODULE_ID is only applicable to iOS MD. For certain HU which supports iPod music play, when HU CarLife switches to background and iPod music plays in the foreground, this command message can be used to pause MD to send Music PCM stream to HU so as to avoid lack of bandwidth in the underlayer. 2. enum ModuleId { CARLIFE_EDOG_MODULE_ID=8; CARLIFE_CRUISE_ID=9; } These two MODULE_IDs are (newly-added PB_VERSION 2.0)	modules and to set related status value
--	--	--	--	---	---



		<pre> _RUNNING =1; } enum ConnectState { CONNECT_STATUS_ADB= 1, CONNECT_STATUS_AOA= 2, CONNECT_STATUS_NCM_ ANDROID=3, CONNECT_STATUS_NCM_ IOS =4, CONNECT_STATUS_WIFI= 5, } enum MicState { MIC_STATUS_USE_VEHIC LE_MIC =0; MIC_STATUS_USE_MOBIL E_MIC =1; } enum MediaPCMState { MEDIAPCM_STATUS_IDLE =0; MEDIAPCM_STATUS_RUN NING=1; } enum EDogState { EDOG_STATUS_IDLE =0; EDOG_STATUS_RUNNING =1; } enum CruiseState { CRUISE_STATUS_IDLE =0; CRUISE_STATUS_RUNNIN G =1; </pre>			
--	--	---	--	--	--



		<pre> } message CarlifeModuleStatus { required int32 moduleID = 1; required int32 statusID = 2; } </pre>			
MSG_C MD_CAR_DATA _GEAR	0x0001 8029	<pre> enum GearState { GEAR_NEUTRAL = 1; GEAR_PARK = 2; GEAR_DRIVE = 3; GEAR_LOW = 4; GEAR_REVERSE = 5; } message CarLifeGearInfo { required GearState gear = 1; } </pre>	HU to MD	The signal of backing a car via gear data. Currently, HU only sends REVERSE gear info; when in other status apart from backing a car, DRIVE info is sent. (This message can be later used to analyze users' driving habit.)	For details, please refer to Chapter Use of HU Data and Suggestions
MSG_C MD_NAVI_NEXT_TURN _INFO	0x0001 0030	<pre> message CarlifeNaviNextTurnInfo { required ActionType action = 1; required TurnKind nextTurn = 2; required string roadName = 3; required int32totalDistance= 4; required int32 remaindistance= 5; required bytesturnIconData= 6; } </pre>	MD to HU	Turn-by-turn navi info sent from MD to HU. This message is sent for displaying related guiding info in other display area out of main display area	Definition of this message and related enumeration can be referred to Chapter13 Navi HUD Info
MSG_C MD_CAR_DATA _SUBSCRIBE	0x0001 0031		MD to HU	Message sent from MD to HU for subscribing car data. This message doesn't contain any content (but to check the type of data this HU supports and the	For details, please refer to Chapter Use of HU Data and Suggestions



				frequency of generating this type of data).	
MSG_C MD_CAR_DATA_SUBSCRIBE_DONE	0x00018032	<pre> enum ModuleID { CAR_DATA_GPS=0; CAR_DATA_VELOCITY=1; CAR_DATA_GYROSCOPE=2; CAR_DATA_ACCELERATION=3; CAR_DATA_GEAR=4; CAR_DATA_OIL=5; } message CarlifeVehicleInfoList { required int32 cnt = 1; repeated com.baidu.carlife.pb.CarLifeVehicleInfo vehicleInfo = 2; } message CarlifeVehicleInfo { required int32 moduleID = 1; required int32 supportFlag = 2; required uint32 frequency = 3; } </pre>	HU to MD	The feedback information for car data subscription sent from HU to MD. supportFlag: check whether this HU supports the car data. (1 means Support while 2 means NOT Support; for GPS_Module, 1 means WGS-84 coordinate system while 2 means GCJ-02 coordinate system) Frequency: frequency of HU generating this kind of data. (When HU doesn't support, set frequency as 0)	For details, please refer to Chapter Use of HU Data and Suggestions
MSG_CMD_CAR_DATA_SUBSCRIBE_START	0x00010033	<pre> enum ModuleID { CAR_DATA_GPS=0; CAR_DATA_VELOCITY=1; CAR_DATA_GYROSCOPE=2; CAR_DATA_ACCELERATION=3; CAR_DATA_GEAR=4; CAR_DATA_OIL=5; } message CarlifeVehicleInfo { required int32 moduleID </pre>	MD to HU	MD requires HU to send subscribed car data. If MD needs to subscribe data of different types, this message shall be sent for multiple times. This message contains only one CarLifeVehicleInfo.	For details, please refer to Chapter Use of HU Data and Suggestions



		<pre> = 1; optional int32 supportFlag = 2; required uint32 frequency= 3; } </pre>			
MSG_C MD_CAR_DATA _SUBSCRIBES TOP	0x0001 0034	<pre> enum ModuleID { CAR_DATA_GPS=0; CAR_DATA_VELOCITY =1; CAR_DATA_GYROSCOPE=2; CAR_DATA_ACCELERATION=3; CAR_DATA_GEAR=4; CAR_DATA_OIL=5; } message CarLifeVehicleInfo { required int32 moduleID = 1; optional int32 supportFlag = 2; optionaluint32 frequency= 3; } </pre>	MD to HU	MD requires HU to stop sending subscribed car data. If MD needs to cancel the subscription of data of different types, this message shall be sent for multiple times. This message contains only one CarLifeVehicleInfo.	For details, please refer to Chapter <u>Use of HU Data and Suggestions</u>
MSG_C MD_MEDIA_INFO	0x0001 0035	<pre> enum CarlifeMediaPlayMode { PLAYMODE_REPEAT_ ONE = 0; PLAYMODE_SHUFFLE = 1; PLAYMODE_REPEAT_ ALL = 2; } message CarlifeMediaInfo { required string source = 1; required string song = 2; required string artist = 3; required string album = 4; required byte albumArt = 5; required int32 duration = 6; required int32 </pre>	MD to HU	Music information sent from MD to HU	This message is sent from MD when the status of music playing has changed. HU will react to this message. Please refer to chapter13.2



		<pre> playlistNum = 7; required string songId = 8; required PlayMode mode = 9; } </pre>			
MSG_C MD_ME DIA_PR OGRES S_BAR	0x0001 0036	<pre> message CarlifeMediaProgressBar { required int32 progressBar = 1; } </pre>	MD to HU	The info on music play progress bar sent from MD to HU(unit:ms)	This message will be sent per second from MD when music is playing
MSG_C MD_CO NNECT_ EXCEPT ION	0x0001 0037	<pre> message CarlifeConnectException { required int32 exceptionType = 1; } TYPE_ENCODER_ERROR = 1; TYPE_PERMISSION_DENIED = 2; TYPE_SCREENSHARE_REQUEST = 3; </pre>	MD to HU	Message indicating connection exception (such as initiation of video encoder fails, or Android5.0 or beyond refuse screen sharing) sent from MD to HU.	TYPE_ENCODER_ERROR indicates exception of encoder TYPE_PERMISSION_DENIED indicates MD users refuse screen sharing TYPE_SCREENSHARE_REQUEST indicates Android 5.0 or beyond is requiring for screen sharing. There will be a pop up on MD. When HU receives this message, it will offer a reminder for users to react.
MSG_C MD_REQUEST_ GO_TO_ _FORE GROUND	0x0001 0038	N/A	MD to HU	Message sent from MD to HU, requiring HU to re-launch CarLife APP on MD	In some cases, HU needs to run command to launch CarLife APP to foreground. At that time, MD needs to send this message



					to HU.
MSG_C MD_UI_ ACTION _SOUN D	0x0001 0039	N/A	MD to HU	MD requires HU to play the keytone voice	N/A
MSG_C MD_BT_ HFP_RE QUEST	0x0001 0040	Enum RequestType { REQUEST_STARTCAL L=1, REQUEST_TERMINAT ECALL=2, REQUEST_ANSWERC ALL=3, REQUEST_REJECTCA LL=4, REQUEST_DTMFCOD E=5, REQUEST_MUTEMIC= 6 } message CarlifeBTHfpRequest { required RequestType command = 1; optional string phonenum = 2; optional int32 dtmfcode = 3; }	MD to HU	Message sent from MD to HU for controlling the BT phone call function on HU	This message is sent for indicating the request for starting phone call when CarLife Internal Telephone UI is used on HU. Details can be referred to Chapter BT Telephone
MSG_C MD_BT_ HFP_IN DICATI ON	0x0001 8041	Enum IndicationType { INDICATION_NEWCALL=1, INDICATION_OUTCALL=2, INDICATION_CALLACTIVE= 3, INDICATION_NOCALL ACTIVE=4 INDICATION_MULTICA LLACTIVE = 5, INDICATION_MULTICA LLINACTIVE = 6 } message CarlifeBTHfpIndication { required IndicationType state = 1; optional string phonenum = 2; optional string name = 3;	HU to MD	Message sent from HU to MD indicating the status of BT phone call on MD. When MD receive Phone status message, it requires HU to fill out BT address field in order to indicate the status source of this BT device. Since some HU support connecting multiple BT devices at the same time.	This message is sent for indicating the status of phone call when CarLife Internal Telephone UI is used on HU. Details can be referred to Chapter BT Phone Call



		optional string address = 4; }			
MSG_CMD_BT_HFP_CONNECTION	0x00018042	<pre> Enum ConnectionState { HFP_DISCONNECTED = 0, HFP_CONNECTING = 1, HFP_CONNECTED = 2 } message CarlifeBTHfpConnection { required ConnectionState state = 1; optional string address = 2; optional string name = 3; } </pre>	HU to MD	Message sent from HU to MD to inform MD of the connection status of BT(HFP)	This message is used for updating BT HFP connection status. When CarLife is connected successfully, HU needs to send this message to inform the MD of the BT HFP connection status. Details can be referred to Chapter BT Phone Call .
MSG_CMD_CARLIFE_DATA_SUBSCRIBE	0x00018043	<pre> enum ModuleID { CARLIFE_DATA_TURN BYTURN=0; CARLIFE_DATA_ASSIS TANTGUIDE=1; } message CarlifeSubscribeMobileCarLifeInfoList { required int32 cnt = 1; repeated com.baidu.carlife.protobuf.CarlifeSubscribeMobileCarLifeInfo = 2; } message CarlifeSubscribeMobileCarLifeInfo { required int32 moduleID = 1; required boolsupportFlag = 2; } </pre>	HU to MD	Message sent from HU to MD for subscribing CarLife data on MD. (Check whether the CarLife version on MD supports providing this type of data) CARLIFE_DATA_TURNBYTURN means turn-by-turn guiding info CARLIFE_DATA_ASSISTANTGUIDE is assistant guiding info	This message is sent when HU needs to use HUD info or cruise info in the navi module of CarLife. After subscription, MD will return the message of MSG_CMD_CARLIFE_DATA_SUBSCRIBE_DONE
MSG_CMD_CARLIFE_FEEDBACK	0x0001	<pre> enum ModuleID { </pre>	MD to HU	The feedback for CarLife data	The feedback for



RLIFE_DATA_SUBSCRIBE_DONE	0044	<pre> CARLIFE_DATA_TURN BYTURN=0; CARLFIE_DATA_ASSIS TANTGUIDE=1; } message CarlifeSubscribeMobileCarLif eInfoList { required int32 cnt = 1; repeated com.baidu.carlife.protobuf.Ca rlifeSubscribeMobileCarLifIn fo = 2; } message CarlifeSubscribeMobileCarLif eInfo { required int32 moduleID = 1; required boolsupportFlag = 2; } </pre>		subscription sent from MD to HU CARLIFE_DATA_TURNBYTURN means turn-by-turn guiding info CARLFIE_DATA_ASSISTANTGUIDE is assistant guiding info supportFlag: whether CarLife version of this MD support providing current data	MSG_CMD_CARLIFE_DATA_SUBSCRIBE sent from HU
MSG_CMD_CARLIFE_DATA_SUBSCRIBE_START	0x00018045	<pre> enum ModuleID { CARLIFE_DATA_TURN BYTURN=0; CARLFIE_DATA_ASSIS TANTGUIDE=1; } message CarlifeSubscribeMobileCarLif eInfoList { required int32 cnt = 1; repeated com.baidu.carlife.protobuf.Ca rlifeSubscribeMobileCarLifIn fo = 2; } message CarlifeSubscribeMobileCarLif eInfo { required int32 moduleID = 1; required boolsupportFlag = 2; } </pre>	HU to MD	HU requires MDs to start sending subscribed info CARLIFE_DATA_TURNBYTURN means turn-by-turn guiding info CARLFIE_DATA_ASSISTANTGUIDE is assistant guiding info	



MSG_C MD_CARLIFE_ DATA_S UBSCRI BE_STO P	0x0001 8046	enum ModuleID { CARLIFE_DATA_TURN BYTURN=0; CARLFIE_DATA_ASSIS TANTGUIDE=1; } message CarlifeSubscribeMobileCarLif eInfoList { required int32 cnt = 1; repeated com.baidu.carlife.protobuf.Ca rlifeSubscribeMobileCarLifIn fo = 2; } message CarlifeSubscribeMobileCarLif eInfo { required int32 moduleID = 1; required boolsupportFlag = 2; }	HU to MD	HU requires MD to stop sending subscribed info. CARLIFE_DATA_T URNBYTURN means turn-by-turn guiding info CARLFIE_DATA_A SSISTANTGUIDE is assistant guiding info	
MSG_C MD_NA VI_ASSI TANTG UIDE_IN FO	0x0001 0047	message CarLifeNaviAssitantGuideInf o { required ActionType action = 1; required AssistantType assistantType =2; required TrafficSignType trafficSignType =3; required int32 totalDistance = 4; required int32 remainDistance = 5; required int32 cameraSpeed = 6; }	MD to HU	The Navi Guiding Info sent from MD to HU. This message is sent for displaying navi guiding info out of main display area.	The definition of this message structure and enumeration can be referred to Chapter 13.
MSG_C MD_HU _AUTHE	0x0001 8048	message CarlifeAuthenRequest { required string	HU to MD	CarLife on HU generates a random character	CarLife on HU conducts Safety



N_REQ UEST		randomValue = 1; }		string, and send it to CarLife on MDs.	Verification of CarLife on MD.
MSG_C MD_MD _AUTHE N_RESP ONSE	0x0001 0049	message CarlifeAuthenResponse { required string encryptValue = 1; }	MD to HU	CarLife on MD encrypts the character string from HU via private key.	CarLife on HU conducts Safety Verification of CarLife on MD.
MSG_C MD_HU _AUTHE N_RES ULT	0x0001 804A	message CarlifeAuthenResult { required bool result = 1; }	HU to MD	CarLife on HU sends the authentication results to CarLife on MD	CarLife on HU conducts Safety Verification of CarLife on MD.
MSG_C MD_MD _AUTHE N_RES ULT	0x0001 004B	message CarlifeAuthenResult { required bool result = 1; }	MD to HU	CarLife on MD sends the authentication results to CarLife on HU.	CarLife on HU conducts Safety Verification of CarLife on MD.
MSG_C MD_GO _TO_FO REGRO UND_R ESPON SE	0x0001 004C	N/A	MD to HU	After HU sends MSG_CMD_GO_T O_FOREGROUND to MD, for switching CarLife to foreground, if MD does not react successfully in 5 seconds, this message will be sent.	N/A
MSG_C MD_ST ART_BT _AUTOP AIR	0x0001 004D	message CarlifeBTStartPairReq { requiredint32 ostype = 1; optional string address = 2; }	MD to HU	This message is sent from MD to HU, using to inform HU to start the BT auto-pair process and to disconnect with current HFP connection (for Android). If this message is sent from an iOS MD, the HU will only need to disconnect with existing HFP connection.	For details please refer to chapter 6.2.3 Auto-pair Initialization



MSG_C MD_BT_ HFP_RE SPONS E	0x0001 804E	Enum RequestType { REQUEST_STARTCAL L=1, REQUEST_TERMINAT ECALL=2, REQUEST_ANSWERC ALL=3, REQUEST_REJECTCA LL=4, REQUEST_DTMFCOD E=5, REQUEST_MUTEMIC= 6 } messageCarlifeBTHfpRespo nse { Required int32 status = 1; Required RequestTypecmd = 2; Optional int32 dtmfCode = 3; }	HU to MD	After HU conducts HFP request, HU will inform MD the result by sending this message.	
MSG_C MD_BT_ HFP_ STATUS_ REQUEST	0x0001 004F	Enum StatusType { STATUS_MIC=1 } message CarlifeBTHfpStatusRequest { Required StatusType type = 1; }	MD to HU	CarLife on MD sends this message to HU, used for obtaining specific status. It mainly used for obtaining HU MIC status.	
MSG_C MD_BT_ HFP_ST_ ATUS_R ESPON SE	0x0001 8050	Enum StatusType { STATUS_MIC=1 } message CarlifeBTHfpStatusResponse { Required int32status = 1; Required StatusType type = 2; }	HU to MD	This message is sent from HU to MD, used for giving response to MD with specific status When HU receives status request from MD. It mainly used for giving response of HU MIC status	
MSG_C MD_MD_ FEATU RE_CO	0x0001 0051	N/A	MD to HU	After passing handshake authentication, MD	



NFIG_R EQUES T				will send this message to HU, requesting HU to conduct customized functionality.	
MSG_C MD_HU _FEATU RE_CO NFIG_R ESPON SE	0x0001 8052	For details, please refer to Table10.2-2	HU to MD	Customized for HU, currently support 6 types of customization, they are microphone (VOICE_MIC), voice wake up (VOICE_WAKEUP), Bluetooth auto pair (BLUETOOTH_AUTO_PAIR), Bluetooth telephone (BLUETOOTH_INTERNAL_UI), focus status (FOCUS_UI), media sample rate (MEDIA_SAMPLE_RATE). There will be more customization message upcoming. Structure of parameter means the PB structure of key value list, sub key of the list is not fixed, HU can use required customization item as key value.	
MSG_C MD_BT_ START_ IDENTIF Y_REQ	0x0001 8053	<pre>message CarlifeBTStartIdentifyReq { Required Stringaddress = 1; }</pre>	HU to MD	HU send this message to make MD start to BT mark process	For details please refer to Chapter 6.2.3 BT auto pairing Initialization
MSG_C MD_BT_ IDENTIF Y_RESU LT_IND	0x0001 0054	<pre>message CarlifeBTIdentifyResultInd { Required intstatus = 1; Required String address = 2; }</pre>	MD to HU	This message is sent to HU from MD, to inform HU the result of BT mark, meanwhile to inform HU when mark status is changed	For details please refer to Chapter 6.2.3 BT auto pairing Initialization



MSG_CMD_ERROR_CODE	0x00018055	message CarlifeErrorCode { required string errorCode = 1; }	HU to MD	This message is sent to MD from HU, indicating the error code	Error code is sent from HU to MD
MSG_CMD_VIDEO_ENCODER_JPEG	0x00018056	N/A	HU to MD	This message is sent from HU to MD for transmitting jpeg data instead of H.264 video data	This message should be sent before HU sends MSG_CMD_VIDEO_ENCODER_INIT
MSG_CMD_VIDEO_ENCODER_JPEG_ACK	0x00010057	N/A	MD to HU	This message is sent from MD to HU, informing that jpeg format is supported	Response will be sent only on the condition that the jpeg format is supported
MSG_CMD_BT_HFP_CALL_STATUS_COVER	0x00010058	message CarlifeCallStatusCover { Required int callStatus = 1; Required String phoneNum = 2; Required String name = 3; }	MD to HU	This message is sent from MD to HU, indicating that the cover should be shown for prompting the calling status	This message is available only when the CarLife Internal UI is supported
MSG_CMD_MD_EXIT	0x00010059		MD to HU	This message is sent to HU before users manually exit CarLife on HU	
Reserved	0x0001005A~0x0001006F				Reserve ID
MSG_CMD_HU_CONNECTION_STATISTIC	0x00018070	message CarlifeConStatisticProto { Required int nConnectTotal = 1; Required int nSuccessCount = 2; Required int nFailedCount = 3; Required int nConnectTime = 4; Required int nErrorCount = 5; RepeatedStringErrorType = 6; }	HU to MD	section statistical message sent to MD by HU nConnectTotal: total count of connection requests initiated by HU or MD nSuccessCount: successful connection and CarLife home page displaying count nFailedCount:	When HU is successfully connected to MD, it will be reported to MD



				failed connection and CarLife home page not successfully displaying count nConnectTime: average time to successfully connect and display the home page ErrorType: error codes indicating that connection failed	
--	--	--	--	--	--

Table 10.2-3 Parameter Configuration of CarLife Fuctionality Customized Message

Key(String)	Value(int)	Purpose
VOICE_MIC	USE_VEHICLE_MIC =0; (Default) USE_MOBILE_MIC =1;	Customized voice record source, using HU mic by default.
VOICE_WAKEUP	ENABLE = 1;(default) DISABLE = 0;	Customized voice wake up, enable by default.
BLUETOOTH_AUTO_PAIR	ENABLE = 1; DISABLE = 0; (default)	Customized Bluetooth auto pair, enable by default
BLUETOOTH_INTERNAL_UI	ENABLE = 1; DISABLE = 0; (default)	Customized Bluetooth telephone, disable by default
FOCUS_UI	ENABLE = 1; DISABLE = 0; (default)	Customized focus status, disable by default.
MEDIA_SAMPLE_RATE	ORIGINAL_SAMPLE_RATE = 0;(default) CUSTOMIZED_SAMPLE_RATE_48K=1;	Customized media sample rate from MD to HU (PCM stream), NOT



		customized by default.
AUDIO_TRANSMISSION_MODE	INDEPENDENT_CHANNEL_MODE=0; (default) BLUE_TOOTH_MODE=1;	Customize the method for transmitting audio stream from MD to HU. It is defaulted to use audio channel to transmit audio
ENGINE_TYPE	FUEL_OIL=0;(default) ELECTRICITY_POWER=1;	Customize HU type running CarLife; default HU type is fuel vehicle
INPUT_DISABLE	INPUT_NOTDISABLE =0; INPUT_DISABLE =1; (default)	For whether to shield input method according to the speed of vehicle, it is defaulted to enable shielding as long as HU sends the car velocity

10.2.3. Sequence Diagram

The Sequence Diagram of some command messages are shown as below:

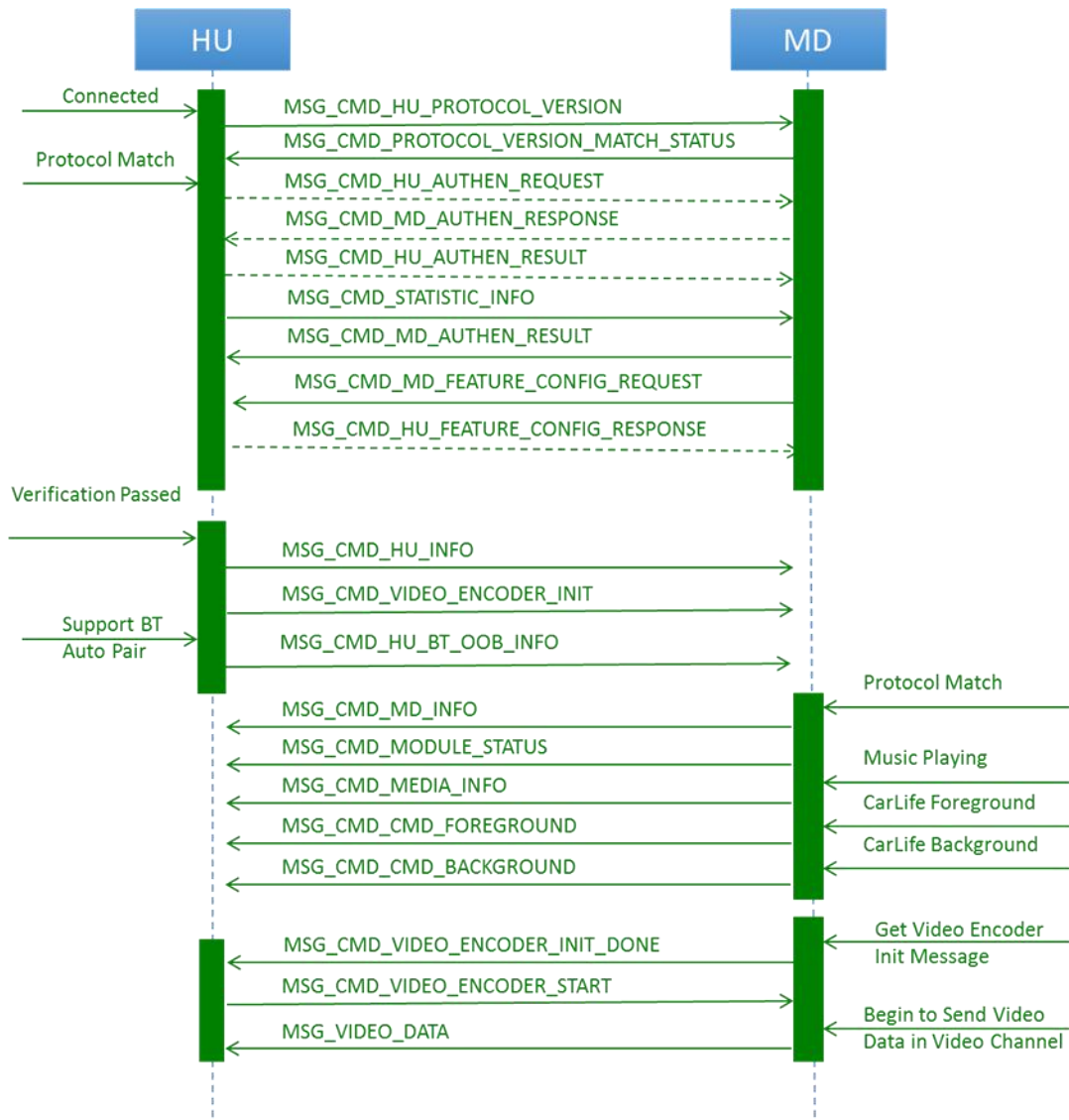


Figure 10.2-2: Sequence Diagram of Some Command Messages

10.3. Video Message

When MD sets up connection with HU, both sides will exchange messages of width and height parameter and frame rate parameter. After some preparations, MD will use H.264 baseline profile to encode and generate video stream, and also start data transmission via TCP socket channel that has been set up. HU should ensure the channel unblocked. On any conditions, it should NOT block writing or reading video data. Also, after receiving video data, HU needs to decode data and display video as soon as possible to avoid delay.

Package structure and Related Messages of video data are illustrated as below:



10.3.1. Package Structure

MD will encode CarLife APP interface to video stream and transmit to HU. HU should decode the video stream and timely display it on HU screen. The CarLife video message structure is shown as below:

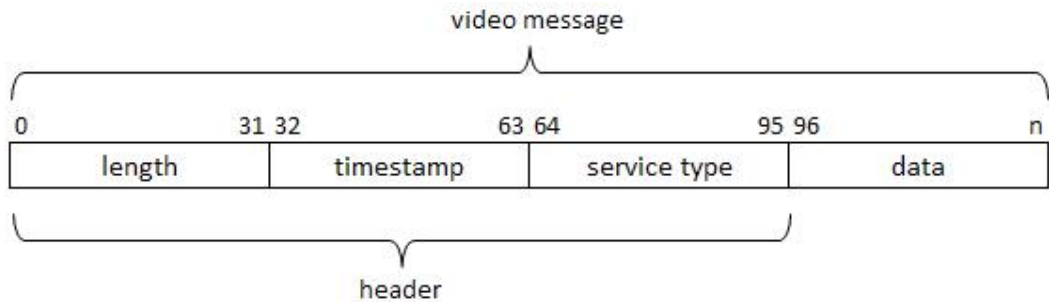


Figure10.3-1: CarLife Video Message Structure

The length of CarLife video message header is fixed as 12 bytes. And the definition of each field is shown as below:

Table10.3-1: Definition of Each Field in CarLife Video Message

Field Names	Offset	Length(bit)	Notes
length	0~31	32	Length of data field
timestamp	32~63	32	Video timestamp, for two purposes: 1. To synchronize with audio. 2. As MD will keep sending this timestamp, the timestamp could be used as benchmark timeline to judge whether audio and video are out of date.
service type	64~95	32	Message service type. The data needed on MD when CarLife stays at background or foreground: 1. When CarLife APP runs at foreground, sending the decoded video data: MSG_VIDEO_DATA (0x00020001); 2. When CarLife runs at background, sending heartbeat package with particular structure: MSG_VIDEO_HEATBEAT (0x00020002). For



			CarLife version beyond CarLifeV4.0, CarLife supports MD background screenshot, so when CarLife runs at background, video data is still sent via MSG_VIDEO_DATA (0x00020001).
data	96~n	length*8	Video data byte stream after encoding.

10.3.2. Related Messages

Video messages are shown as below:

Table 10.3-2:

Message	Service type	PB Structure of Data Field	Direction	Definition & Functions	Scenario
MSG_CMD_VIDEO_ENCODER_INIT	0x00018007	<pre> message CarLifeVideoEncoderInfo { required int32 width = 1; required int32 height = 2; required int32 frameRate = 3; } </pre>	HU to MD	Request for video engine initiation sent from HU to MD.	Video interaction message for the first time when connection is set up.
MSG_CMD_VIDEO_ENCODER_DONE	0x00010008	<pre> message CarLifeVideoEncoderInfo { required int32 width = 1; required int32 height = 2; required int32 frameRate = 3; } </pre>	MD to HU	Message sent from MD to HU indicating that initiation of video engine has been successfully done, and returning resolution and frame rate parameter confirmed by MD to HU.	This message is sent from MD to HU when initiation of video engine has successfully done.



MSG_CMD_VIDEO_ENCODE_START	0x00018009	N/A	HU to MD	Message sent from HU to MD indicating video data transmission starts.	This message is sent after HU receives MSG_CMD_VIDEO_ENCODE_INIT_DONE and starts to receive video data; or when video pausing status resumes to video displaying status on HU.
MSG_CMD_VIDEO_ENCODE_PAUSE	0x0001800A	N/A	HU to MD	Message sent from HU to MD indicating video transmission pauses.	This message is sent when HU does NOT need to display video or when the video displaying on HU pauses due to other reasons.
MSG_CMD_VIDEO_ENCODE_FRAME_RATE_CHANGE	0x0001800C	message CarLifeVideoFrameRate { required int32 frameRate = 1; }	HU to MD	Request sent from HU to MD for changing video frame rate	This message is sent when the maximum frame rate needs to be changed on HU.
MSG_CMD_VIDEO_ENCODE_FRAME_RATE_CHANGE_DONE	0x0001000D	message CarLifeVideoFrameRate { required int32 frameRate = 1; }	MD to HU	Message sent from MD to HU indicating that video frame rate has successfully changed,	This message is sent from MD to HU when adjustment of maximum frame rate has successfully done on MD.



				and returning the frame rate after adjustment.	
--	--	--	--	--	--

The parameter field of each message can be referred to the following sections:

10.3.2.1. MSG_CMD_VIDEO_ENCODER_INIT

Field Name	Type	Optional/ Required	Description
width	int32	Required	The width of display area on HU, in pixels.
height	int32	Required	The height of display area on HU, in pixels.
frameRate	int32	Required	The frame rate desirable for HU, frames per second

Table 10.3-3

10.3.2.2. MSG_CMD_VIDEO_ENCODER_INIT_DONE

Field Name	Type	Optional/Required	Description
width	int32	Required	The resolution width confirmed by MD, in pixels.
height	int32	Required	The resolution height confirmed by MD, in pixels.
frameRate	int32	Required	The frame rate confirmed by MD, frames per second

Table 10.3-4



10.3.2.3.

MSG_CMD_VIDEO_ENCODER_FRAME_RATE_CHANGE

Field Name	Type	Optional/ Required	Description
frameRate	int32	Required	The maximum frame rate desirable for HU, frames per second

Table 10.3-5

10.3.2.4.

MSG_CMD_VIDEO_ENCODER_FRAME_RATE_CHANGE_DONE

Field Name	Type	Optional/ Required	Description
Frame rate	int32	Required	The maximum frame rate confirmed by MD, frames per second

Table 10.3-6

10.3.3 Sequence Diagram

Message interaction depicted as below is through Command Message Channel, and is not related to Video Data Channel.

10.3.3.1 Video Message Sequence Diagram When Setting Up Connection

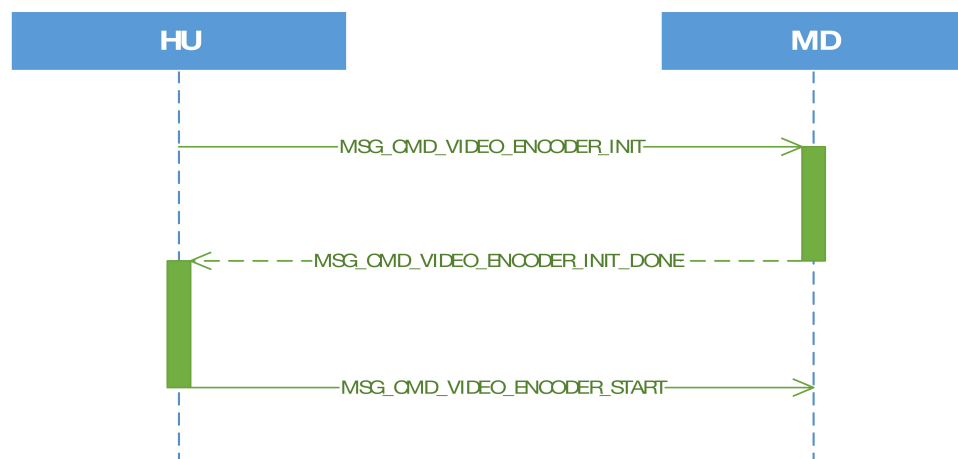


Figure 10.3-2

After connection is set up, HU should actively send message of `MSG_CMD_VIDEO_ENCODER_INIT` to MD to initialize encoder via command message channel. The message should include the width and height parameter of video and should set a limit to the maximum frame rate parameter that is desirable for HU. Then, wait MD complete the initiation of encoder and reply message of `MSG_CMD_VIDEO_ENCODER_INIT_DONE` to inform HU of completing initiation. Here, the message should include the width and height parameter and the maximum frame rate parameter that have been confirmed by MD. After HU receive this message and create encoder according to those parameters, it is able to send the message of `MSG_CMD_VIDEO_ENCODER_START`. After that, MD can generate video data, and send data to HU via TCP socket channel that has been particularly set up for video data transmission.

10.3.3.2. Video Message Sequence Diagram When Adjusting Frame Rates

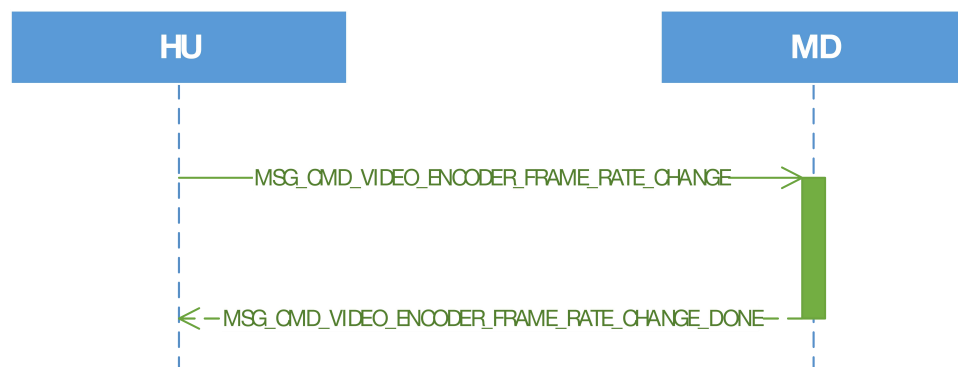


Figure 10.3-3

When HU needs to increase or decrease video frame rates, HU needs to send message of `MSG_CMD_VIDEO_ENCODER_FRAME_RATE_CHANGE` to MD via Command Message Channel. The message should contains the parameter of target frame rate. When MD receives message, MD checks the feasibility of those parameters and makes adjustment, MD will return a confirmation message of `MSG_CMD_VIDEO_ENCODER_FRAME_RATE_CHANGE_DONE`, which contains the frame rate after being adjusted by MD.

10.3.3.3. Video Message Sequence Diagram When Pausing Video Data Transmission

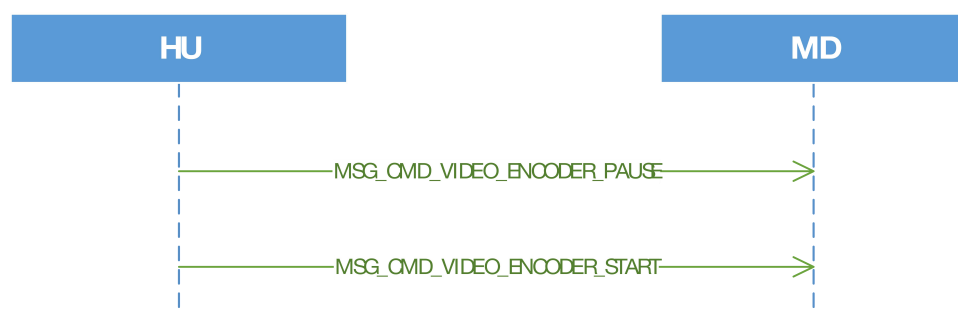


Figure 10.3-4

When HU turns off screen, or switches to background and does not display video any more, HU need to notify MD of pausing the generation of video data. HU send



MSG_CMD_VIDEO_ENCODER_PAUSE via Command Message Channel; after MD receives this message, video engine will stop generating video data, and only send heartbeat package to remain connection.

When HU resumes displaying video images from screen turn-off or switch to run at background, HU should send MSG_CMD_VIDEO_ENCODER_START via Command Message Channel to notify MD of resuming video engine.

10.4. Audio Message

MD can encode the audio data in CarLife and transmit data to HU via TCP socket. HU should parse the audio data and decide how to play this audio data.

10.4.1. Package Structure

CarLife audio message structure is shown as below:

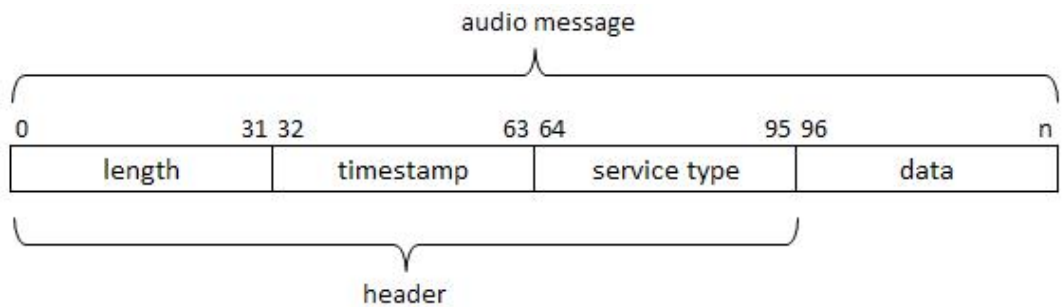


Figure 10.4-1: CarLife Audio Message Structure

The length of CarLife audio message header is fixed as 12 bytes, and the definition of each field is shown as below:

Field	Offsets	Length(bit)	Field Attribute and Meaning
length	0~31	32	The length of data field.
timestamp	32~63	32	Timestamp for synchronizing with video data.
service type	64~95	32	Message service type, the status of audio data transmission on MD. For Media audio, service types are defined as:



			initiation, data transmission, pause, resume playing. For TTS audio, service types are defined as: initiation, data transmission and stop data transmission.
data	96~n	length*8	If service type is data transmission, this field is for encoded audio data byte (little endian). If it is other service type, this field is for byte stream with parameters serialized by PB.

Table 10.4-1: Field Definition of CarLife Audio Data Message

10.4.2. Related Messages

1. Media Audio Message Definition

For Media message, except that data field of MSG_MEDIA_DATA is the original PCM stream, data fields of other messages are serialized by PB. Definition of service type and corresponding data field on MD is shown as below:

Table 10.4-2: Service Type and Data of CarLife Media Audio Message

Field Macro Definition	service type Value	data	Direction	Meaning and Function	Scenario
MSG_MEDIA_INIT	0x00030001	message CarLifeMediaInit { required int32 sampleRate = 1; required int32 channelConfig = 2; required int32 sampleFormat = 3; }	MD to HU	Media initiation on MD, which mainly includes sampling rate, channel configuration, quantization accuracy.	This message is sent when starting music play.
MSG_MEDIA_PAUSE	0x00030003	N/A	MD to HU	Media on MD pauses.	This message is sent when music play is paused.
MSG_MEDIA_RESUME_PLAY	0x00030004	N/A	MD to HU	Media on MD resumes playing.	This message is sent when music play is resumed



MSG_MEDIA_DATA	0x00030006	Media Audio Stream in PCM format	MD to HU	Media audio data sent from MD.	This message is sent when transmitting music data.
----------------	------------	----------------------------------	----------	--------------------------------	--

Table 10.4-3: MSG_MEDIA_INIT Data Field Definition

Field Name	Type	Optional/ Required	Description
sampleRate	int32	Required	Sampling rate(Hz)
channelConfig	int32	Required	Channel numbers (mono track -1, or dual track -2)
sampleFormat	int32	Required	Quantization Format (bits)

2. Navi TTS Audio Message Definition

For Navi TTS message, except that the data field of MSG_NAVI_TTS_DATA is the original PCM stream, the data fields of other messages are all serialized by PB. The definition of service type and data field on MD is shown as below.

Table 10.4-4: Service Type and Data of CarLife Navi TTS Audio Message

Field Macro Definition	service type Value	data	Direction	Definition and Function	Scenario
MSG_NAVI_TTS_INIT	0x00040001	message CarLifeTTSInit { required int32 sampleRate = 1; required int32 channelConfig =2; required int32 sampleFormat =3; }	MD to HU	Navi TTS Initiation on MD, which includes: sampling rate, channel configuration, quantization accuracy.	This message is sent when starting to play Navi TTS.
MSG_NAVI_TTS_END	0x00040002	N/A	MD to HU	Navi TTS on MD stops.	This message is sent when stopping Navi TTS play.
MSG_NAVI_TTS_DATA	0x00040003	TTS Audio Stream in PCM	MD to HU	Navi TTS audio data sent from MD.	TTS data is transmitted when playing Navi TTS.

Table 10.4-5: MSG_NAVI_TTS_INIT Data Field Definition

Field Name	Type	Optional/ Required	Description
------------	------	--------------------	-------------



sampleRate	int32	Required	Sampling rate (Hz)
channelConfig	int32	Required	Channel numbers (mono track -1, or dual track -2)
sampleFormat	int32	Required	Quantization Format(bits)

10.4.3. Sequence Diagram

- MD sends Navi TTS data to HU, which is shown as below:

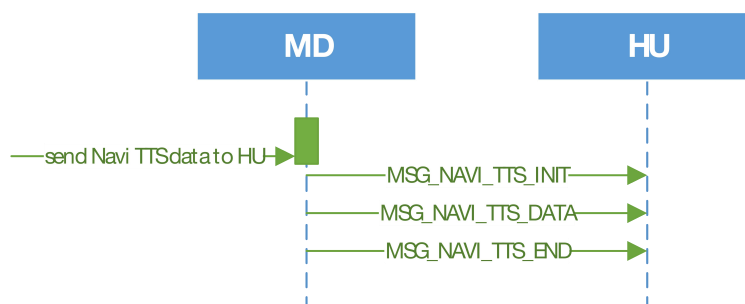


Figure 10.4-2 Navi TTS Message Sequence Diagram

Before MD sends Navi TTS data, MD will firstly send MSG_NAVI_TTS_INIT which contains audio initiation parameters and then sends audio data (marked as MSG_NAVI_TTS_DATA). When MD stops sending Navi TTS data, MD will send MSG_NAVI_TTS_END to notify HU.

- MD sends Media data to HU, which is shown as below:

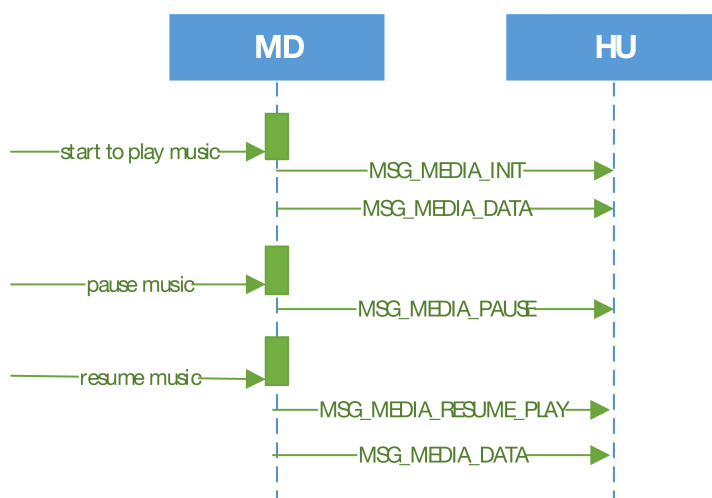


Figure 10.4-3 Media Message Sequence Diagram

When music on MD starts to play, MD will send MSG_MEDIA_INIT which contains audio initiation message to HU, and then send audio data (marked as MSG_MEDIA_DATA);

When music play on MD pauses, MD will send MSG_MEDIA_PAUSE to notify HU;

When music on MD resumes playing, MD will send MSG_MEDIA_RESUME_PLAY to notify HU that music has already resumed playing, and meanwhile, send audio data (marked as MSG_MEDIA_DATA).

3. HU controls Navi Stop, VR Stop, Media Pause and Resume on MD via Command Channel, which is shown as below:

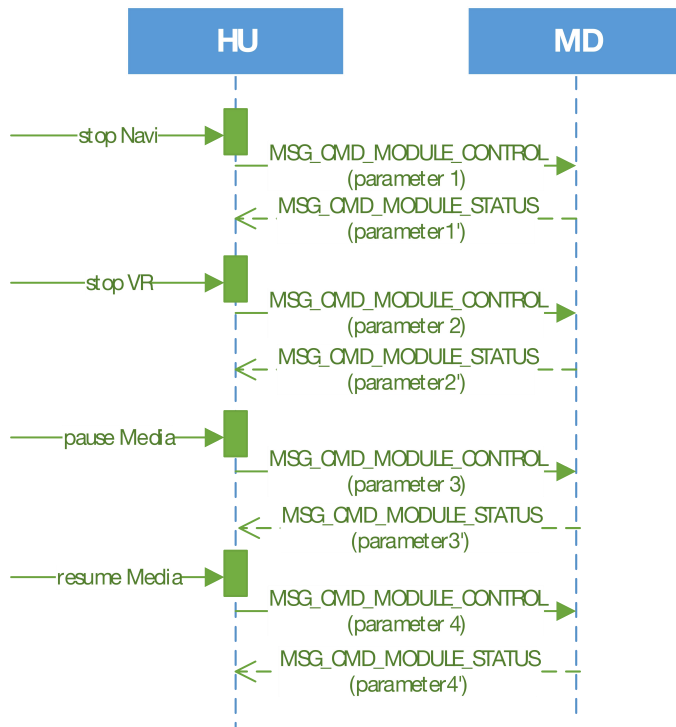


Figure 10.4-4 Sequence Diagram of Audio Controlling via Command Channel

HU controls the stop of Navi, VR, Media Module via MSG_CMD_MODULE_CONROL command (at the same time, it can control Media Resume). When the status of corresponding modules on MD changes, MD will feedback the module change info to HU via MSG_CMD_MODULE_STATUS.



10.5. Voice Message

10.5.1. Package Structure

CarLife voice message will be sent to HU from MD via VR channel while HU transmits recorded audio data (via MIC) to MD via VR channel. CarLife voice message structure is shown as below:

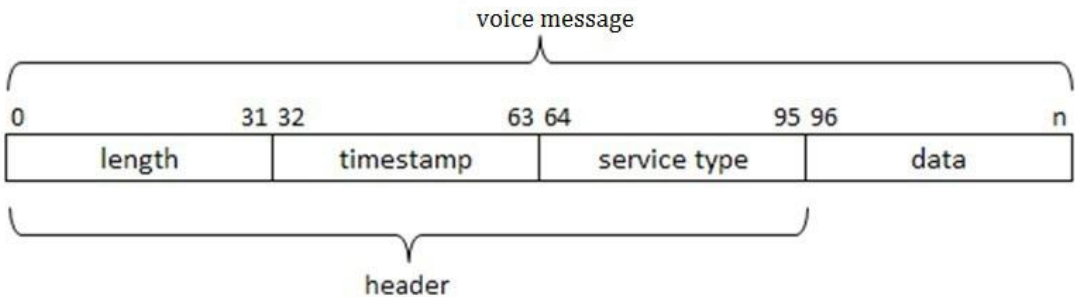


Figure 10.5-1: CarLife Voice Message Structure

The length of CarLife voice message header is fixed as 12 bytes, and the definition of each field is shown as below:

Field Name	Offsets	Length(bit)	Field Attribute and Meaning
length	0~31	32	The length of data field (network byte order).
timestamp	32~63	32	Timestamp information which can be ignored (network byte order).
service type	64~95	32	Service type of voice data information (network byte order).
data	96~	1024 recommended	This field indicates the real transmitted voice record data, which shall be 16Bit depth ((little endian), 16K sampling rate and mono channel.

Table 10.5-1: Field Definition of CarLife Voice Data Message



10.5.2. Related Messages

Table 10.5-2 displays all messages Carlife needs to run its VR function. Those messages transmitted via Command Channel should be packed by PB while voice data messages transmitted via VR Channel need NOT be packaged as VR Channel is used to transmit data stream; if messages are packaged by PB, it will waste time for compressing and extracting.

Table 10.5-2 Definition of CarLife Command Message Service Type and Data

Message	Service type	Data	Direction	Definition and Function	Scenario
MSG_CMD_MIC_RECORD_WAKEUP_START	0x00010022	N/A	MD to HU	This message is sent via Command Channel from MD to HU, which notifies HU of starting Voice Wakeup. If MSG_CMD_HU_FEATURE_CONFIG_RESPONSE sets up to use HU MIC, when HU receives this command message, HU shall compress data stream to packages (1kb per package), and send them to MD one by one. For details of sending, please refer to MSG_VR_DATA. If MSG_CMD_HU_FEATURE_CONFIG_RESPONSE sets up to use MD MIC, MD is responsible for recording.	This message is sent from MD when voice wakeup starts
MSG_CMD_MIC_RECORD_END	0x00010023	N/A	MD to HU	This message is sent via Command Channel from MD to HU, requesting HU to pause recording.	This message is sent from MD when voice wakeup or voice



					recognition ends.
MSG_CMD_MIC_RECORD_START	0x00010024	N/A	MD to HU	This message is sent via Command Channel from MD to HU, which informs HU of entering voice recognition and starting recording. If MSG_CMD_HU_FEATURE_CONFIG_RESPONSE sets up to use HU MIC, when HU receives this command message, HU shall compress data stream to packages (1kb per package), and send them to MD one by one. For details of sending, please refer to MSG_VR_DATA. If MSG_CMD_HU_FEATURE_CONFIG_RESPONSE sets up to use MD MIC, MD is responsible for recording.	This message is sent from MD when voice recognition starts.
MSG_VR_DATA	0x00058001	This is the voice recording data that has been actually transmitted. For the quality of voice data, please refer to above depictions. The size of each data package is 1kb, and the data	HU to MD	This message which contains voice PCM stream recorded by MIC, is sent via VR Channel from HU to MD. For the message structure, please refer to the package structure mentioned above.	This message is sent from HU when HU sends voice recording data to MD.



		packages do NOT need to be encapsulated by PB			
MSG_VR_AUDIO_INIT	0x00050002	message CarLifeTTSInit { required int32 sampleRate = 1; required int32 channelConfig = 2; required int32 sampleFormat = 3; }	MD to HU	This message is transmitted via voice channel. VR TTS Initiation on MD mainly includes: sampling rate, channel configuration, quantization accuracy. For the message structure, please refer to the package structure mentioned above.	If VR TTS generates during voice recognition process, this message is sent from MD when VR TTS broadcast starts playing each time.
MSG_VR_AUDIO_DATA	0x00050003	VR TTS audio stream in PCM format	MD to HU	This message is sent via VR Channel from MD to HU which contains VR TTS audio data. For the message structure, please refer to the package structure mentioned above.	This message is sent from MD when VR TTS broadcast starts playing. The message contains real-time VR TTS data.
MSG_VR_AUDIO_STOP	0x00050004	N/A	MD to HU	This message is sent via VR Channel when VR TTS on MD stops playing. For the message structure, please refer to the package structure mentioned above.	This message is sent from MD indicating the VR TTS playing ends. Notes: this message is not VR TTS interrupt message; it only means that the



					current VR TTS has come to the end. For VR TTS interrupt message, please refer to MSG_VR_AUDIO_INTERRUPT
MSG_COMMAND_MODULE_STATUS	0x00010026	<pre>enum ModuleId { CARLIFE_PHONE_MODULE_ID=1; CARLIFE_NAVI_MODULE_ID=2; CARLIFE_MUSIC_MODULE_ID=3; CARLIFE_VR_MODULE_ID=4; // 将 RECORD 关键字修改为 VR CARLIFE_CONNECT_MODULE_ID=5; CARLIFE_MIC_MODULE_ID=6; CARLIFE_MEDIA_PCM_MODULE_ID=7; CARLIFE_EDOG_MODULE_ID=8; (P/S: PB_VERSION 2.0 新增) CARLIFE_CRUISE_ID=9; (P/S: PB_VERSION 2.0 新增) } enum PhoneState</pre>	MD to HU	This message is transmitted via Command Channel to synchronize the status when MD and HU are connected. And this message can be sent from MD to HU indicating the status of different modules (music, voice, Navi) has changed. Notes: 1. CARLIFE_MEDIAPCM_MODULE_ID is only applicable to iOS MD. For certain HU which supports iPod music play, when HU CarLife switches to background and iPod music plays in the foreground, this command message can be used to pause MD to send Music PCM stream to HU so as to avoid lack of bandwidth in the	This message is sent from MD to HU when status of music, Navi and VR changes.



		<pre> { PHONE_STATUS _IDLE=0; PHONE_STATUS _INCOMING=1; PHONE_STATUS _OUTING=2; } enum NaviState { NAVI_STATUS_I DLE =0; NAVI_STATUS_R UNNING =1; } enum MusicState { MUSIC_STATUS _IDLE=0; MUSIC_STATUS _RUNNING=1; } enum VRState { VR_STATUS_ RECORD _IDLE =0; VR_STATUS_ RECORD _RUNNING =1; } enum ConnectState { CONNECT_STAT US_ADB=1, CONNECT_STAT US_AOA=2, CONNECT_STAT US_NCM_ANDR OID=3, CONNECT_STAT </pre>	underlayer. 2. <pre> enum ModuleId { CARLIFE_EDOG_M ODULE_ID=8; CARLIFE_CRUISE _ID=9; } </pre> These two MODULE_ID are (new-added PB_VERSION 2.0).	
--	--	--	--	--



		<pre>US_NCM_IOS =4, CONNECT_STAT US_WIFI=5, } enum MicState { MIC_STATUS_U SE_VEHICLE_MI C =0; MIC_STATUS_U SE_MOBILE_MIC =1; } enum MediaPCMState { MEDIAPCM_STA TUS_IDLE=0; MEDIAPCM_STA TUS_RUNNING= 1; } enum EDogState { EDOG_STATUS_ IDLE =0; EDOG_STATUS_ RUNNING =1; } enum CruiseState { CRUISE_ STATUS_IDLE =0; CRUISE_ STATUS_RUNNI NG =1; }</pre>			
--	--	---	--	--	--



		<pre> message CarlifemoduleStat usList { required int32 cnt = 1; repeated com.baid u.carlife.protobuf. CarlifemoduleStat us moduleStatus = 2; } message CarlifemoduleStat us { required int32 moduleID = 1; required int32 statusID = 2; } </pre>			
MSG_CM D_MODULE E_CONTR OL	0x0001 8028	<pre> enum ModuleId { CARLIFE_PHON E_MODULE_ID= 1; CARLIFE_NAVI_ MODULE_ID=2; CARLIFE_MUSIC _MODULE_ID=3; CARLIFE_VR_M ODULE_ID=4; // 将 RECORD 关键 字修改为 VR CARLIFE_CONN ECT_MODULE_I D=5; CARLIFE_MIC_M ODULE_ID=6; CARLIFE_MEDIA </pre>	HU to MD	This message is transmitted via Command Channel to synchronize the status when MD and HU are connected: US can control different modules (music, voice, Navi) on MD via this message. Notes: CARLIFE_MEDIAPC M_MODULE_ID is only applicable to iOS MD. For certain HU which supports iPod	This message is sent to MD when HU needs to turn on/off music, telephone and Navi. This is also a way for non-Android MD to have HardKey control.



		<pre> PCM_MODULE_ID=7; CARLIFE_EDOG_MODULE_ID=8; (P/S: PB_VERSION 2.0 新增) CARLIFE_CRUISE_ID=9; (P/S: PB_VERSION 2.0 新增) } enum PhoneState { PHONE_STATUS_IDLE=0; PHONE_STATUS_INCOMING=1; PHONE_STATUS_OUTING=2; } enum NaviState { NAVI_STATUS_IDLE =0; NAVI_STATUS_RUNNING =1; } enum MusicState { MUSIC_STATUS_IDLE=0; MUSIC_STATUS_RUNNING=1; } enum VRState { VR_STATUS_RECORD_IDLE=0; VR_STATUS_RECORD </pre>		music play, when HU CarLife switches to background and iPod music plays in the foreground, this command message can be used to pause MD to send Music PCM stream to HU so as to avoid lack of bandwidth in the underlayer. 2. <pre> enum ModuleId { CARLIFE_EDOG_MODULE_ID=8; CARLIFE_CRUISE_ID=9; } </pre> These two MODULE_ID are (new-added PB_VERSION 2.0).	
--	--	--	--	--	--



		<pre> _RUNNING =1; } enum ConnectState { CONNECT_STAT US_ADB=1, CONNECT_STAT US_AOA=2, CONNECT_STAT US_NCM_ANDR OID=3, CONNECT_STAT US_NCM_IOS =4, CONNECT_STAT US_WIFI=5, } enum MicState { MIC_STATUS_U SE_VEHICLE_MI C =0; MIC_STATUS_U SE_MOBILE_MIC =1; } enum MediaPCMState { MEDIAPCM_STA TUS_IDLE=0; MEDIAPCM_STA TUS_RUNNING= 1; } enum EDogState { EDOG_STATUS_ IDLE =0; EDOG_STATUS_ RUNNING =1; </pre>			
--	--	--	--	--	--



		<pre>} enum CruiseState { CRUISE_STATU S_IDLE =0; CRUISE_STATU S_RUNNING =1; } message CarlifeModuleStat us { required int32 moduleID = 1; required int32 statusID = 2; }</pre>			
MSG_VR_MODULE_STATUS	0x00050005	Similar to MSG_CMD_MODULE_STATUS	MD to HU	This message substitutes for MSG_CMD_MODULE_STATUS which synchronizes VR status. This is because: when VR is activated, MD will send MSG_CMD_MODULE_STATUS to inform HU that VR has been activated, and then send MSG_VR_AUDIO_INIT to HU, requesting for palying “greetings”. As these 2 messages are sent via different channels, it is very likely that	To ensure the compatibility, MSG_CMD_MODULE_STATUS will still be sent to HU, while MSG_VR_MODULE_STATUS will only be sent to certain type of HU. Only one message shall be applied.



				MSG_VR_AUDIO_INTERRUPT, which is sent later, might arrive on HU first; in this case, some HU will deal with this as exceptions. In order to fundamentally solve this problem, MSG_VR_MODULE_STATUS, which is sent via VR Channel, is added to replace MSG_CMD_MODULE_STATUS so as to guarantee sequence conformity.	
MSG_VR_AUDIO_INTERRUPT	0x00050006	None	MD to HU	If recognition status is activated again during VR broadcast process, MD will send this message to HU; when HU receives this message, it needs to stop the current VR broadcast and wipe VR TTS cache.	When VR is playing a longer VR TTS documents (such as jokes), if VR status needs to be activated again, it requires stopping the current VR TTS broadcast. To do this, MD will first send MSG_VR_AUDIO_STOP to indicate that the current VR TTS has



					come to the end, and then send MSG_VR_AUDIO_INTERRUPT to stop broadcasting current VR TTS.
--	--	--	--	--	--

Notes: it requires attention for HU to process VR channel messages. When receiving VR channel messages, HU shall instantly judge the service type; if the service type is MSG_VR_AUDIO_INTERRUPT, HU shall instantly discard unplayed VR data in the buffer.

10.5.3. Sequence Diagram

10.5.3.1. Voice Wakeup Sequence



Figure 10.5-2: The Voice Wakeup Interactive Sequence Diagram

- 1) When voice **wakeup** starts, MD will send `MSG_CMD_MIC_RECORD_WAKEUP_START` to HU, notifying HU that voice wakeup function has been activated and HU should turn on audio-recording thread.
- 2) HU will continuously send `MSG_VR_DATA` to MD to transmit voice data package (1KB per package).
- 3) When VR engine on MD identifies “小度小度(Xiao Du, Xiao Du)”, it will end voice wakeup for this time, and send **`MSG_CMD_MIC_RECORD_END`** to HU, notifying HU to end audio recording and then start another VR process.

10.5.3.2. Voice Recognition Sequence

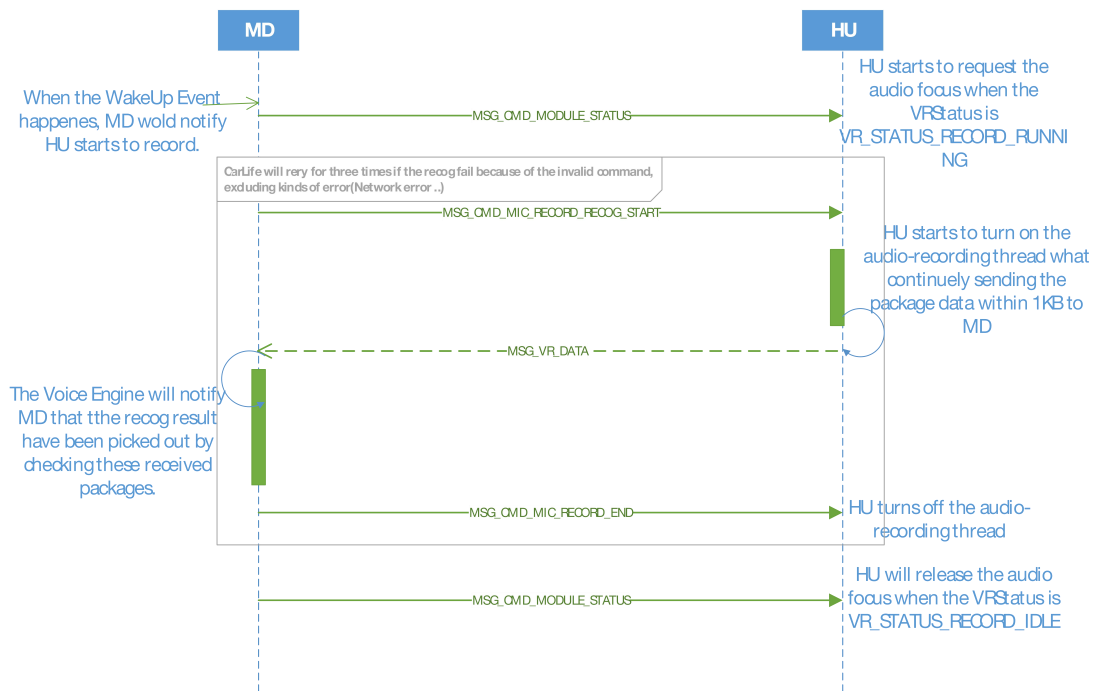


Figure 10.5-3: The Voice Recognition Interactive Sequence Diagram

- 1) When mask layer of voice recognition is being activated, MD will send MSG_CMD_MODULE_STATUS to HU first, notifying HU to launch its VR; HU needs to apply for audio focus; meanwhile, MD will initiate VR engine;
 - 2) After VR engine is activated and ready, MD will send MSG_CMD_MIC_RECORD_RECOG_START, which requires HU activating MIC recording thread; then HU will continuously send MSG_VR_DATA to MD to transmit audio data package (1KB per package);
 - 3) When MD has recognition results returned from VR engine, MD will send MSG_CMD_MIC_RECORD_END, notifying HU to end MIC recording thread now;
 - 4) When mask layer of voice recognition has been activated, MD will start a new round of mask layer of voice recognition of VR via MSG_CMD_MIC_RECORD_RECOG_START message only; MSG_CMD_MODULE_STATUS (parameter value VR_STATUS_RECORD_RUNNING) will not be sent again;
- Notes: MSG_CMD_MODULE_STATUS (parameter value VR_STATUS_RECORD_RUNNING) will only be sent when mask layer of voice recognition is being activated.
- 5) When mask layer of voice recognition is being turned off, MD will stop VR engine and send MSG_CMD_MODULE_STATUS (parameter value VR_STATUS_RECORD_IDLE) to HU; after receiving this message, HU will release audio focus. Besides, MD will send MSG_CMD_MIC_RECORD_END, notifying HU to end recording thread and stop sending MSG_VR_DATA message to MD.

10.5.3.3. Audio Source Customization Sequence

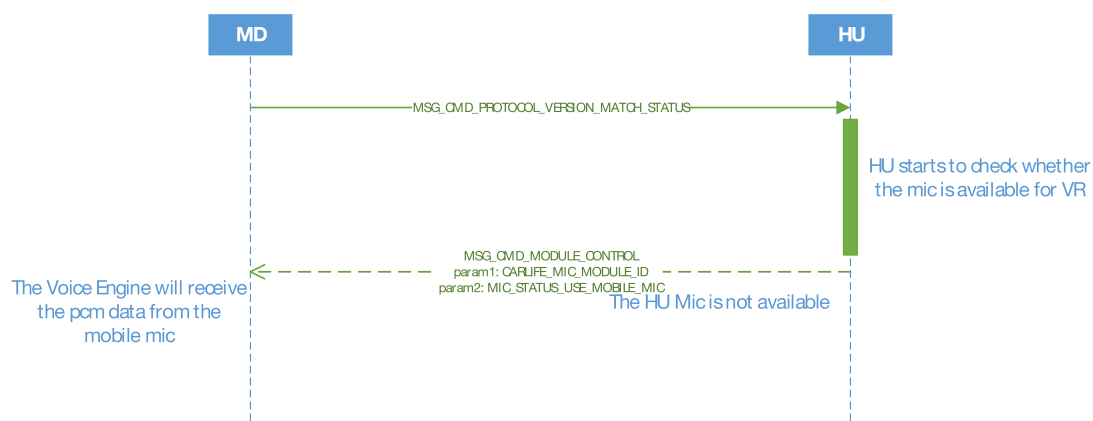


Figure 10.5-4 MIC Customization and Interaction Sequence Diagram

VR and Voice Wakeup can be implemented by both HU MIC and MD MIC. If not specifically required, HU MIC shall be implemented by default.

If the recording quality of HU MIC cannot meet the requirements of VR engine, and MD MIC is considered, 2 methods can be applied, among which the Method 1 is recommended for configuration:

- 1) When HU sends `MSG_CMD_HU_FEATURE_CONFIG_RESPONSE` to MD, parameter value of `VOICE_MIC` shall be set as `USE_MOBILE_MIC`. For details, please refer to the definition of `MSG_CMD_HU_FEATURE_CONFIG_RESPONSE`;
- 2) `MSG_CMD_MODULE_CONTROL` can also be sent; parameter value of `ModuleId` is `CARLIFE_MIC_MODULE_ID` while that of `MicState` is `MIC_STATUS_USE_MOBILE_MIC`. The message shall be sent after HU receives `MSG_CMD_PROTOCOL_VERSION_MATCH_STATUS`.

10.5.3.4. VR-TTS Broadcast Sequence

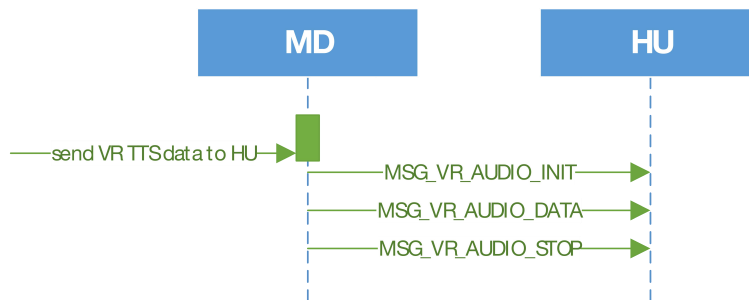


Figure 10.5-5 VR TTS Broadcast Sequence Diagram

- 1) During VR process, certain VR TTS data will be generated to offer auditory feedback to users. Before MD sends the VR TTS data, it shall send MSG_VR_AUDIO_INIT which contains audio initiation parameters first;
- 2) Then MD will continuously send real-time VR TTS audio data marked as MSG_VR_AUDIO_DATA to HU via VR Channel;
- 3) When VR TTS is fully played or interrupted, MD will send MSG_VR_AUDIO_STOP to HU, indicating this VR TTS has ended;
- 4) When playing a longer VR TTS documents (such as jokes, Encyclopedia), if a new round of VR needs to be activated, MD will first send MSG_VR_AUDIO_STOP to indicate that the current VR TTS has come to the end, and then send MSG_VR_AUDIO_INTERRUPT to HU to stop broadcasting current VR TTS, and then enter Step 2 and Step 3 of 10.5.3.2 Recognition Sequence;
Notes: if recording thread is activated when HU VR TTS hasn't been fully stopped, part of this VR TTS will be recorded and sent to HU as recording data, resulting in false recognition.
- 5) When mask layer of voice recognition is being turned off, MD will send MSG_VR_AUDIO_STOP and MSG_VR_AUDIO_INTERRUPT for once; HU shall handle the messages according to requirements to ensure VR TTS has been fully handled.

10.6. Back Control Messages

10.6.1. Back Control Event Strategy

10.6.1.1. Touch Screen

If the interface of HU is touch screen, For MD, there are two ways to implement back control, that is, transparent method and non-transparent method; the transparent method is more



direct and responsive, so it is recommended to use.

10.6.1.1.1. Transparent Method

The touch events consist of ACTION_DOWN, ACTION_UP and ACTION_MOVE. When HU screen captures touch events of users, it will send the captured three events to MD for response and recognition. This is the transparent method, and the event sequence can be seen as follows:

- Single click event: ACTION_DOWN to ACTION_UP
- Double click event: ACTION_DOWN to ACTION_UP to ACTION_DOWN to ACTION_UP
- Long press and release event: ACTION_DOWN to ACTION_MOVE to ACTION_MOVE to ... to ACTION_UP
- Long press and drag event : ACTION_DOWN to ACTION_MOVE to ... to ACTION_UP
- Drag event: ACTION_DOWN to ACTION_MOVE to ACTION_MOVE to ... to ACTION_UP

10.6.1.1.2. Coordinate Transformation

For touch screen, if the resolutions of MD and HU are different, the resolutions of video transmitted are also different. Therefore, the coordinates of touch event on HU should be transformed to corresponding coordinates on MD. Transformation rules are as follows:

- (1) The coordinate on HU is (x0, y0), and the area for displaying video on HU is (w0, h0), the resolution of video stream after receiving MSG_CMD_VIDEO_ENCODER_INIT_DONE is (w1, h1), then the coordinate of pb structure of each event is (x1, y1). The formula is shown as follows:

$$x1 = (\text{int}) (x0 * w1 / w0);$$

$$y1 = (\text{int}) (y0 * h1 / h0);$$

- (2) The real coordinate of touch event on MD is (x2,y2), the resolution of video stream is (w1, h1), the resolution of Mobile Device is (w2,h2). The formula is shown as follows:



$$x2 = (\text{int}) (x1 * w2 / w1);$$

$$y2 = (\text{int}) (y1 * h2 / h1);$$

10.6.1.2. Hard Keys

In order to let users control MD Apps from Hard Keys on HU and improve user experience, a Hard Keys message is added in Back Control Channel sent from HU to MD. According to positions of these hard keys, the scenarios consist of HU, Steering Wheel and Knob, only applicable when HU CarLife runs at foreground. HU can choose to support certain keys. When HU captures a certain Hard Key event, HU will encapsule the corresponding keycode values to PB structure and send this message to MD. For details of key code values and the descriptions, please refer to the following table.

Table 10.6-1: CarLife Hard Keys Message

Hard key name	key Code	Description	Notes	Scenario
KEYCODE_Home	0x00000001	Home key	Click the Home key on MD.	HU
KEYCODE_Phone_Call	0x00000002	Phone call	Use CarLife on MD to make or answer phone call.	HU, steering wheel
KEYCODE_Phone_End	0x00000003	Hang up	Use CarLife on MD to hang up phone call.	HU, steering wheel
KEYCODE_Phone_End_Mute	0x00000004	Hang up, mute media	Use CarLife on MD to hang up phone call and mute media.	Abandoned
KEYCODE_HFP	0x00000005	Switch from hand free mode and MD phone	Switch from hand free mode and MD phone	HU
KEYCODE_Selector_Next	0x00000006	Switch to next focus	Switch to next focus in current focus area	Knob
KEYCODE_Selector_Prev	0x00000007	Switch to previous focus	Switch to previous focus in current focus area	Knob



Previous	00007	previous focus	current focus area	
KEYCODE_Setting	0x00000008	Enter system setting page	Enter CarLife setting page	Abandoned
KEYCODE_Media	0x00000009	Switch to Media page	Access Music module in CarLife on MD	HU
KEYCODE_Radio	0x0000000A	Switch to Radio Page	Enter Radio page in CarLife	Abandoned
KEYCODE_Navi	0x0000000B	Switch to Navi page	Access Music module in CarLife in Navi module	HU
KEYCODE_SRC	0x0000000C	Switch Radio and Media	Switch to Radio and Media page in CarLife	Abandoned
KEYCODE_Mode	0x0000000D	Switch Media	Switch to media playing source in CarLife	Abandoned
KEYCODE_Back	0x0000000E	Back to upper level page	Go back to last page in CarLife app	HU
KEYCODE_Seek_Sub	0x0000000F	Last song	Play last song	HU
KEYCODE_Seek_Add	0x00000010	Next song	Play next song	HU
KEYCODE_Volume_Sub	0x00000011	Reduce volume	Reduce volume on MD	Abandoned
KEYCODE_Volume_Add	0x00000012	Increase volume	Increase volume on MD	Abandoned
KEYCODE_Mute	0x00000013	Mute	Mute during phone call	HU, to implement mute
KEYCODE_OK	0x00000014	Confirm button	When focus selected, press that button as 'click'	HU
KEYCODE_Move_Left	0x0000	Focus area	Switch to the left focus area beside current focus are.	Knob



	00015	move left	After switching, focus will stay in the last focus in this area.	
KEYCODE_Move_Right	0x00000016	Focus area move right	Switch to the right focus area beside current focus area. After switching, focus will stay in the last focus in this area.	Knob
KEYCODE_Move_Up	0x00000017	Focus area move up	Switch to the top focus area beside current focus area. After switching, focus will stay in the last focus in this area.	Knob
KEYCODE_Move_Down	0x00000018	Focus area move down	Switch to the down focus area beside current focus area. After switch, focus will stay in the last focus in this area.	Knob
KEYCODE_Move_Up_Left	0x00000019	Move to upper left	Only support in CarLife Map module	Knob
KEYCODE_Move_Up_Right	0x0000001A	Move to upper right	Only support in CarLife Map module	Knob
KEYCODE_Move_Down_Left	0x0000001B	Move to lower left	Only support in CarLife Map module	Knob
KEYCODE_Move_Down_Right	0x0000001C	Move to lower right	Only support in CarLife Map module	Knob
KEYCODE_TEL	0x0000001D	Switch to phone page	Access phone module in CarLife app	HU
KEYCODE_MAIN	0x0000001E	Switch to home page	Access home page in CarLife app (first page)	HU
KEYCODE_Media_Start	0x0000001F	Play music	Play current music	HU



KEYCODE_Media_Stop	0x00000020	Pause music	Pause current music	HU
KEYCODE_VR_Start	0x00000021	Launch VR	Activate voice recognition, and show VR screen in CarLife	HU, and steering (support long press)
KEYCODE_VR_Stop	0x00000022	Quit VR	Deactivate Voice recognition and, quite the VR page	HU, and steering (support long press)
KEYCODE_NUMBER_0	0x00000023	Numeric keys 0	Numeric keys 0 on dialing key pad	HU
KEYCODE_NUMBER_1	0x00000024	Numeric keys 1	Numeric keys 1 on dialing key pad	HU
KEYCODE_NUMBER_2	0x00000025	Numeric keys 2	Numeric keys 2 on dialing key pad	HU
KEYCODE_NUMBER_3	0x00000026	Numeric keys 3	Numeric keys 3 on dialing key pad	HU
KEYCODE_NUMBER_4	0x00000027	Numeric keys 4	Numeric keys 4 on dialing key pad	HU
KEYCODE_NUMBER_5	0x00000028	Numeric keys 5	Numeric keys 5 on dialing key pad	HU
KEYCODE_NUMBER_6	0x00000029	Numeric keys 6	Numeric keys 6 on dialing key pad	HU
KEYCODE_NUMBER_7	0x0000002A	Numeric keys 7	Numeric keys 7 on dialing key pad	HU
KEYCODE_NUMBER_8	0x0000002B	Numeric keys 8	Numeric keys 8 on dialing key pad	HU
KEYCODE_NUMBER_9	0x0000002C	Numeric keys 9	Numeric keys 9 on dialing key pad	HU



KEYCODE_NUMBER _10	0x000 0002D	Character keys *	Charater keys * on dialing key pad	HU
KEYCODE_NUMBER _11	0x000 0002E	Character keys #	Charater keys # on dialing key pad	HU
KEYCODE_NUMBER _DEL	0x000 0002F	Delete key	Delete a character in dial number input box	HU
KEYCODE_NUMBER _CLEAR	0x000 00030	Clear key	Clear all strings in dial number input box	HU
KEYCODE_NUMBER _PLUS	0x000 00031	+ key	When Long press* key MD, send this Keycode	HU

10.6.1.2.1. Hard Keys in Map Page

Add drag mode button on CarLife' map homepage. Once CarLife successfully connects with HU, Map Drag Mode will be activated in CarLife app, and user can access this mode by clicking the Map Drag Mode button. In this mode, other elements in map module will be hided as full screen, and original cursor will be modified as another type. This cursor will stay in the screen center. Users can drag the map and zoom in/out. Available actions include: Push knob to drag; rotate knob clockwise to zoom in; rotate knob counterclockwise to zoom out. In this mode, pressing the knob will have no response. After entering Map Drag Mode, users choose to switch to other pages; Map Drag Mode will be ended when they return to Map page again. Press Back on HU, users can quit Map Drag Mode and the focus will return to the default nearby icon on the Map page.

10.6.3. Package Structure

CarLife Back Control Messages is transmitted via Touch Channel, which mainly transmits the Back Control event commends captured on HU to MD. CarLife Back Control Messages structure is the same with Command Message structure, which is shown as follows:

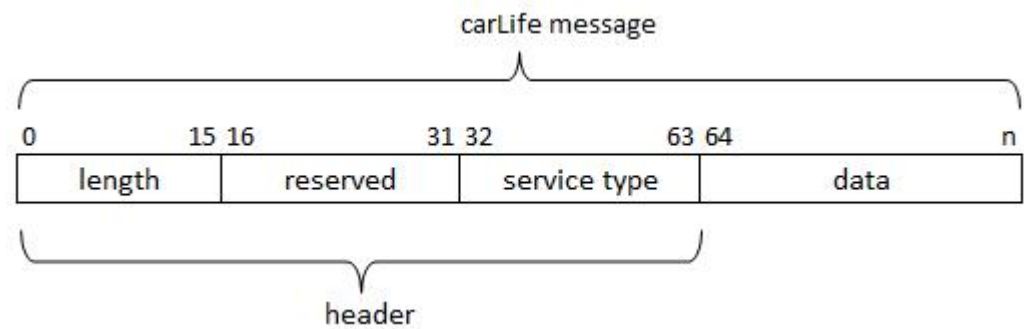


Figure 10.6-1: CarLife Back Control Messages Structure

The length of CarLife Back Control Messages structure is fixed as 8 bytes. Definition of each field is shown as below:

Table 10.6-2: Field Definition of CarLife Back Control Messages

Field	Offset	Length(bit)	Notes
length	0~15	16	Length of data field which is serialized by PB
reserved	16~31	16	Reserved field
service type	32~63	32	Message service type
data	64~n	length*8	The parameter of service tpe field, the byte stream serialized by PB

10.6.4. Related Messages

Data fields of all Back Control Messages service type are serialized by PB. Definitions of service types and corresponding data field on HU and MD are shown as follows:

Table 10.6-3: CarLife Back Control Service Types and Data Field Definitions

MSG_TOUCH_ACTION	0x00 0680 01	message CarlifeTouchAction { required int32 action = 1; required int32 x = 2; required int32 y = 3; }	Touch events sent from HU to MD. Touch events ACTION_DOWN, ACTION_UP and ACTION_MOVE are set as 0, 1, 2 respectively. For calculation of x and y, please refer to Section Coordinates .
------------------	--------------------	--	---



			<u>Transformation.</u> (This is message is applicable to all transparent methods)
MSG_TOUCH_CAR_HARD_KEY_CODE	0x00068008	message CarLifeCarHardKeyCode { required int32 keycode = 1; }	HARD_KEY_CODE event sent from HU to MD. Keycode refers to the serial number of specific event. For details, please refer to Section <u>Hard Keys</u> .

10.6.5. Sequence Diagram

The Sequence Diagram of transparent method on touch screen is shown as below:

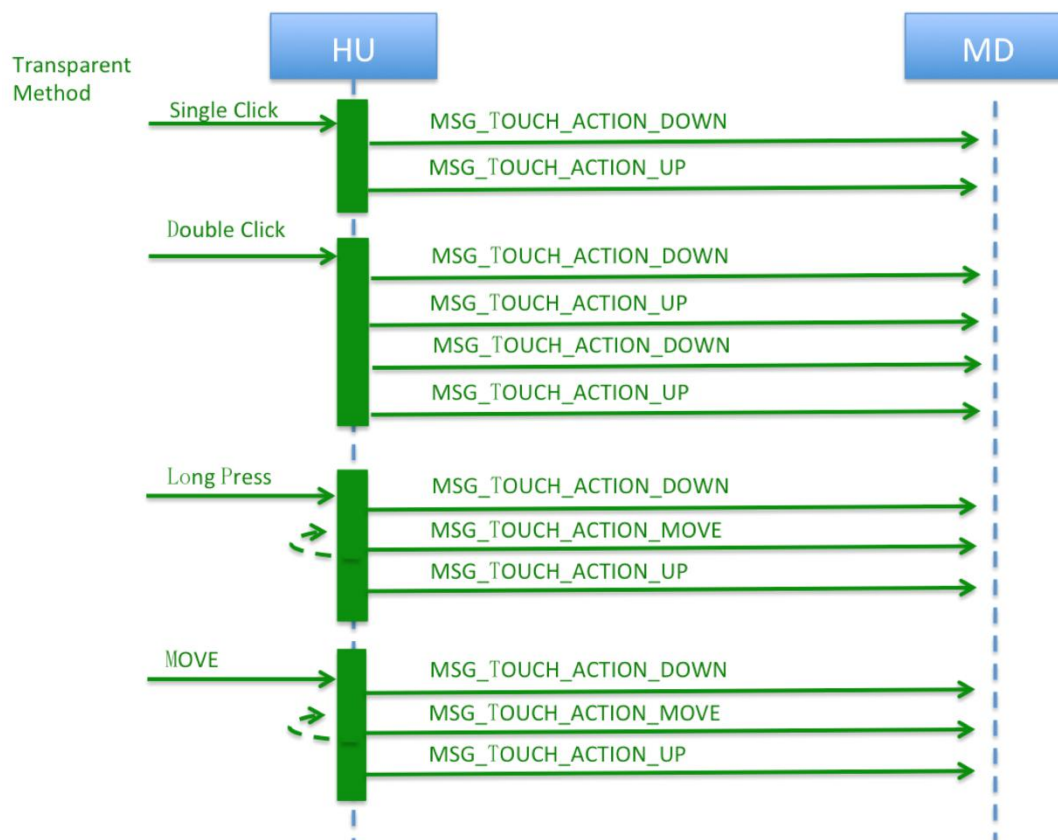


Figure 10.6-2: Sequence Diagram of Touch Screen Transparent Method

The Sequence Diagram of non-transparent method on touch screen is shown as below:

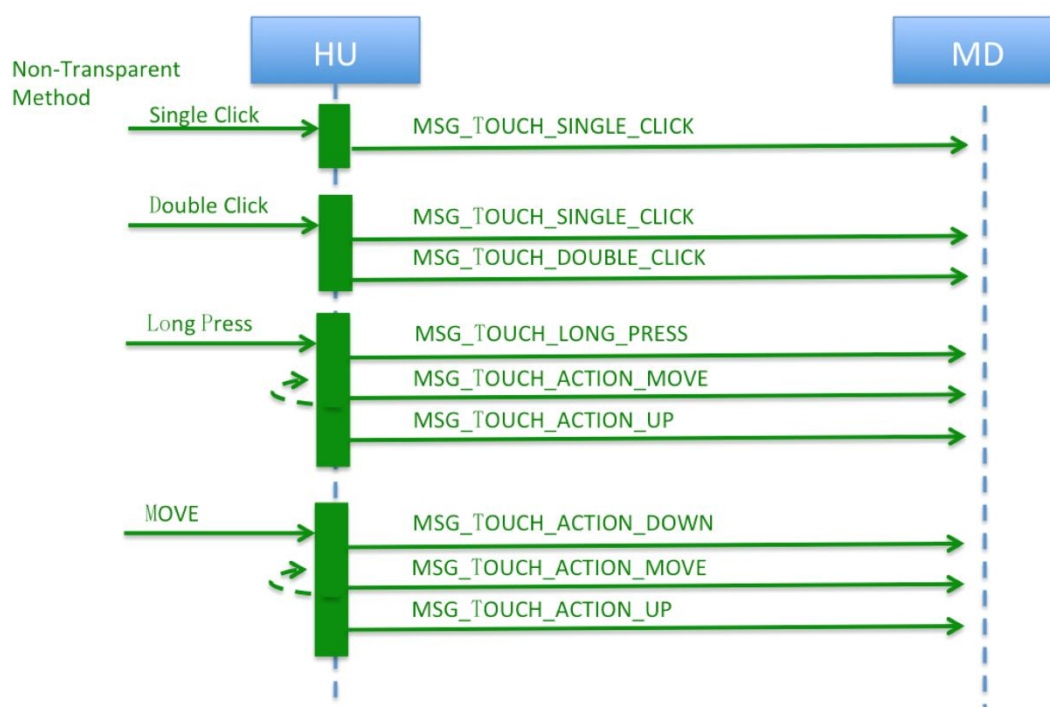


Figure 10.6-3: Sequence Diagram of Touch Screen Non-transparent Method

The Sequence Diagram of physical hard keys is shown as below:

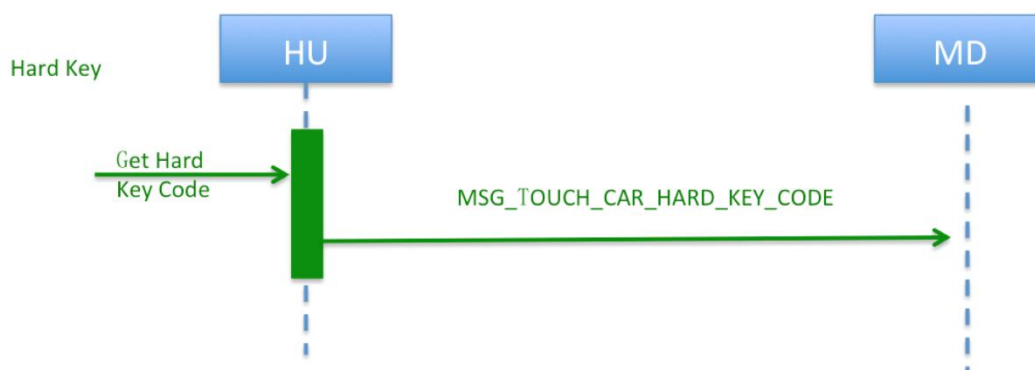


Figure 10.6-4: Sequence Diagram of Physical Hard Keys

11. CarLife on MD Special Scene Strategy

This chapter will focus on describing CarLife on MD's strategy regarding to locked screen, background and phone calls, also involved how HU should deal with these scenarios.

11.1. Screen Lock and Background/Foreground Switch in CarLife APP

The CarLife versions (Android: CarLife 3.0 and beyond; iOS: CarLife 4.0 and beyond) for the new CarLife video solution (APP screenshot solution) support MD background or screen lock. Therefore, under normal circumstances, neither iOS nor Android MD will send background message when MD app runs in background.

However, due to iOS system restrictions, sometimes background screenshot encoding might have error; in this case, MD will send MSG_CMD_BACKGROUND to stop sending video stream data package and start sending video heartbeat package. When HU receives MSG_CMD_BACKGROUND, it must switch to the prompt page when CarLife runs in the background. For details of the prompt page, please refer to Section [CarLife HMI Design](#).

When CarLife on MD switches from background to foreground, it will stop sending video heartbeat package and resume the transmission of video stream data package, and send protocol message MSG_CMD_FOREGROUND to HU via command channels. When HU receives this message, it switches from background running prompt page to video page, and starts decoding and playing video again.

It's similar with the Android App which doesn't support background screenshot.

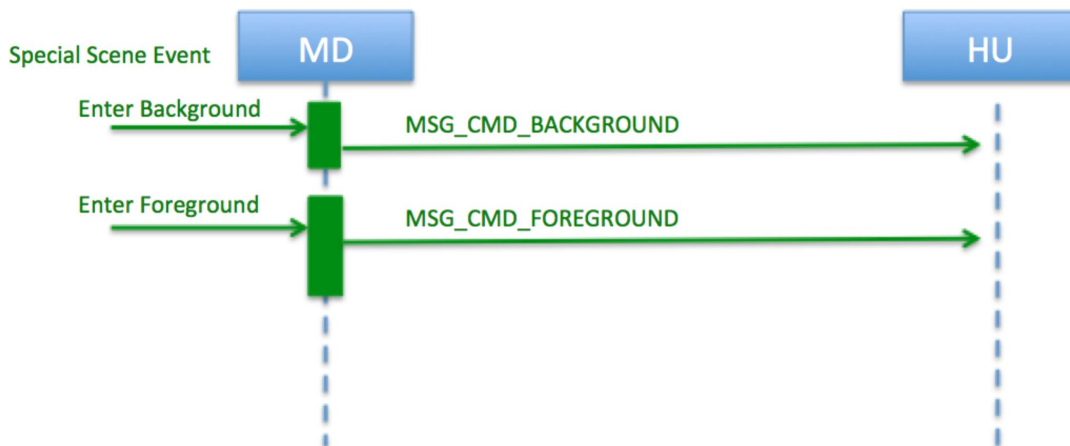


Figure 11.1-1

11.2. MD Phone Call Process

If HU supports CarLife' APP Bluetooth Telephone function, and MD and HU have already been paired and connected, please refer to Chapter [Bluetooth Telephone](#) for CarLife telephone.



If HU doesn't support CarLife' APP Bluetooth Telephone function, or MD & HU haven't been connected via BT, CarLife on MD will inform HU of phone call status so that HU can offer prompt message on the Phone Call Status Prompt Page. For the design of Phone Call Status Prompt Page, please refer to Chapter HU HMI Design.

When the phone call status on MD changes, MD APP will send MSG_CMD_MODULE_STATUS message to inform HU of the current status; at this time, the module ID of this message is the phone module, statusID respectively are:

- When the phone call ends, statusID is PHONE_STATUS_IDLE; after receiving this message, HU will switch back from the phone call status prompt page to the video play page.
- When there is a new incoming call, statusID is PHONE_STATUS_INCOMING; after receiving this message, HU will switch to the phone call status prompt page, which will display a prompt for the incoming call.
- When users are dialing or in a phone call, statusID is PHONE_STATUS_OUTGOING; after receiving this message, HU will switch to the phone call status prompt page, which will display the a prompt for the undergoing call.

12. Use of HU Data and Suggestions

CarLife can obtain vehicle related data from HU and **transmit back to MD** so as to improve MD data and user experience. At present, data which can be obtained from HU includes: vehicle velocity, GPS positioning information, Gyroscope acceleration information and fuel consumption. **Data back-transmission shall follow data subscription mechanism** with the communication protocols as follows:

Related Messages of **vehicle data** are listed as the Table below:

Table 12-1: Service Type and Data Field Definition of CarLife Vehicle Data Message

Name of message	Service type	Data	Direction of flow	Definition & Function	Scenes
MSG_CMD_CAR_VELOCITY	0x0001800F	message CarLifeCarSpeed { required uint32 speed = 1; optional uint64 timeStamp = 2; }	HU to MD	Current car speed sent from HU to MD	Used in CarLife navigation.
MSG_CM	0x00	message CarLifeCarGps	HU	Current GPS	Used in



D_CAR_GPS	018010	<pre> { required uint32 antennaState = 1; // dBHz required int32 latitude = 3; // 1/1,000,000 degrees required int32 longitude = 4; // 1/1,000,000 degrees required int32 height = 5; // 0.1 meters required uint32 speed = 6; // 0.01m/s units required uint32 heading = 7; // 0.1 degrees // units 0.1 required uint32 pdop = 16; // units 0.1 required uint32 satsUsed = 18; required uint32 satsVisible = 19; required uint32 timeStamp = 25; } </pre>	to MD	position sent from HU to MD. Supports WGS84 and GCJ02 coordinate systems. The default transmission frequency is 1Hz.	CarLife navigation.
MSG_CMD_CAR_GYROSCOPE	0x00018011	<pre> enum GyroType { SINGLE_AXIS_GYRO = 1; THREE_AXIS_GYRO = 2; } message CarLifeGyroscope { required GyroType gyroType = 1; required doublegyroX = 2; required doublegyroY = 3; required doublegyroZ = 4; optional uint64 timeStamp = 5; } </pre>	HU to MD	Current car gyroscope info sent from HU to MD. For HU with single-axis gyroscope, set gyroX as the value of single-axis gyroscope, and gyroY and gyroZ as 0. The value of gyroscope is angular velocity (degree/second). gyroX: the angle	Used in CarLife navigation.



				between the projection and the north direction of the HU in Y-axis horizontal plane. The default transmission frequency is 10Hz.	
MSG_CMD_CAR_ACCELERATION	0x00018012	<pre>message CarLifeAcceleration { required doubleaccX = 1; required doubleaccY = 2; required doubleaccZ = 3; optional uint64 timeStamp = 4; }</pre>	HU to MD	Current car acceleration info sent from HU to MD. The default transmission frequency is 10Hz.	Used in CarLife navigation.
MSG_CMD_CAR_OIL	0x00018013	<pre>message CarLifeOil { required int level = 1; required int range = 2; required bool lowFuleWarning = 3; }</pre>	HU to MD	Current fuel consumption info sent from HU to MD.	CarLife reminds users to refuel by displaying fuel consumption information.
MSG_CMD_CAR_DATA_GEAR	0x00018029	<pre>enum GearState { GEAR_NEUTRAL = 1; GEAR_PARK = 2; GEAR_DRIVE = 3; GEAR_LOW = 4; GEAR_REVERSE = 5; } message CarLifeGearInfo { required GearState gear = 1; }</pre>	HU to MD	Mark reverse signal via gear info. In this case, HU only sends REVERSE gear; HU sends DRIVE gear for non-reverse situations.	CarLife uses this information to distinguish the current driving status and forbid or activate certain functions accordingly.
MSG_CMD_CAR_DATA_SUBSCRIBE	0x00010031		MD to HU	Subscribed car data info sent from MD to HU; this message contains no content. (Check	CarLife subscribed car data info



				data type this HU supports and frequency of different types of car data)	
MSG_CM D_CAR_D ATA_SUB SCRIBE_ DONE	0x00 0180 32	<pre>enum ModuleID { CAR_DATA_GPS= 0; CAR_DATA_VELO CITY=1; CAR_DATA_GYR OSCOPE=2; CAR_DATA_ACC ELERATION=3; CAR_DATA_GEA R=4; CAR_DATA_OIL=5 ; } message CarlifeVehicleInfoList { required int32 cnt = 1; repeated com.baidu.ca rlife.protobuf.CarLifeV ehicleInfovehicleInfo = 2; } message CarLifeVehicleInfo { required int32 moduleID = 1; required int32 supportFlag = 2; required uint32 frequency= 3; }</pre>	HU to MD	The feedback info of subscribed car data from HU to MD. supportFlag: Whether this HU supports the current car data. (1 indicates YES while 0 indicates NO; for GPS_Module, 1 indicates WGS-84 coordinate system while 2 indicates GCJ-02 coordinate system. Frequency: The frequency of HU generating this type of data. (Frequency is set as 0 when this HU doesn't support the car data)	CarLife subscribed car data info
MSG_CM D_CAR_D ATA_SUB SCRIBE_S TART	0x00 0100 33	<pre>enum ModuleID { CAR_DATA_GPS= 0; CAR_DATA_VELO CITY=1; CAR_DATA_GYR OSCOPE=2; CAR_DATA_ACC</pre>	MD to HU	MD requires HU to send subscribed car data. If MD needs to subscribe data of different types, this message shall be sent for multiple times. This	CarLife subscribed car data info



		<pre> ELERATION=3; CAR_DATA_GEA R=4; CAR_DATA_OIL=5 ; } message CarLifeVehicleInfo { required int32 moduleID = 1; required int32 supportFlag = 2; required uint32 frequency= 3; } </pre>		message contains only one CarLifeVehicleInfo.	
MSG_CM D_CAR_D ATA_SUB SCRIBE_S TOP	0x00 0100 34	<pre> enum ModuleID { CAR_DATA_GPS= 0; CAR_DATA_VELO CITY=1; CAR_DATA_GYR OSCOPE=2; CAR_DATA_ACC ELERATION=3; CAR_DATA_GEA R=4; CAR_DATA_OIL=5 ; } message CarLifeVehicleInfo { required int32 moduleID = 1; required int32 moduleID = 1; required int32 supportFlag = 2; required uint32 frequency= 3; } </pre>	MD to HU	MD requires HU to stop sending subscribed car data. If MD needs to cancel the subscription of data of different types, this message shall be sent for multiple times. This message contains only one CarLifeVehicleInfo.	CarLife subscribed car data info



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

The detailed field definitions will be described in the following sections.

12.1. MSG_CMD_CAR_VELOCITY

The field definition of MSG_CMD_CAR_VELOCITY is shown in the following table:

Field Name	Type	Optional/ Required	Description
speed	int32	Required	Current speed of vehicle, Unit: km/h
TimeStamp	int64	Optional	The timestamp when HU sends this message, Unit: ms

12.2. MSG_CMD_CAR_GPS

The field definition of MSG_CMD_CAR_VELOCITY is shown in the following table:

Field Name	Type	Optional/ Required	Description
antennaState	int32	Required	The status of GPS module: -1 indicates Unavailable; 1 indicates Available.
latitude	int32	Required	Latitude information, Unit: 1/1,000,000 degrees
longitude	int32	Required	Longitude information, Unit: 1/1,000,000 degrees
height	int32	Required	Altitude information, Unit: 1/10 m
speed	int32	Required	Speed information Unit: 1/100m/s
heading	int32	Required	Vehicle direction, Unit: 1/10 degrees
pdop	int32	Required	Position accuracy factor, Unit: 1/10 accuracy
satsUsed	int32	Required	The number of satellites used for positioning
statsVisible	int32	Required	The number of satellites that have been searched
timeStamp	int64	Required	The timestamp when HU sends this message, Unit:ms

12.3. MSG_CMD_CAR_GYROSCOPE

The field definition of MSG_CMD_CAR_GYROSCOPE is shown in the following table:



Field Name	Type	Optional/ Required	Description
gyroType	GyroType	Required	enum GyroType <pre> { SINGLE_AXIS_GYRO = 1; THREE_AXIS_GYRO = 2; } </pre> The type of Gyroscope. To set gyroX be the value of the single-axis gyroscope, the value of gyroY and gyroZ is 0 when Head Unit uses the single-axis gyroscope.
gyroX	int32	Required	The angle between the projection and the north direction of the HU in Y-axis horizontal plane. Unit: degrees / sec. The default transmission frequency is 10Hz.
gyroY	int32	Required	
gyroZ	int32	Required	
timeStamp	int64	Required	The timestamp when HU sends this message, Unit:ms

12.4. MSG_CMD_CAR_ACCELERATION

The field definition of MSG_CMD_CAR_ACCELERATION is shown in the following table:

Field Name	Type	Optional/ Required	Description
accX	double	Required	Horizontal-direction acceleration. Right is positive. Unit: m/s ² .
accY	double	Required	Horizontally vertical acceleration The front is positive. Unit: m/s ²
accZ	double	Required	Spatially vertical acceleration Sky direction is positive Unit: m/s ²
timeStamp	int64	Required	The timestamp when HU sends this message, Unit:ms



12.5. MSG_CMD_CAR_OIL

The field definition of MSG_CMD_CAR_OIL is shown in the following table:

Field Name	Type	Optional/ Required	Description
level	int32	Required	Percent of current oil (0-100)
range	int32	Required	The distance that the car can drive (Unit:km)
lowFuleWarning	bool	Required	Whether the oil is too low.

12.6. MSG_CMD_CAR_DATA_GEAR

The field definition of MSG_CMD_CAR_DATA_GEAR is shown in the following table:

Field Name	Type	Optional/ Required	Description
gear	Int32	Required	The current gear status: <pre>enum{ GEAR_NEUTRAL = 1; GEAR_PARK = 2; GEAR_DRIVE = 3; GEAR_LOW = 4; GEAR_REVERSE = 5; }</pre> Among which, GEAR_NEUTRAL: NEUTRAL; GEAR_PARK: PARK; GEAR_DRIVE: DRIVE; GEAR_LOW: LOW SPEED; GEAR_REVERSE: REVERSE;

12.7. MSG_CMD_CAR_DATA_SUBSCRIBE

The field definition of MSG_CMD_CAR_DATA_SUBSCRIBE is:

MD shall check all data types the HU supports to subscribe; the message contains no content.

12.8. MSG_CMD_CAR_DATA_SUBSCRIBE_DONE

The field definition of MSG_CMD_CAR_DATA_SUBSCRIBE_DONE is shown in the following table:

Field Name	Type	Optional/ Required	Description
cnt	int32	Required	The number of data modules



			this HU supports to subscribe, vehicleInfo field will be several sets, the number of which is cnt.
vehicleInfo	message CarLifeVehicleInfo	Required	Message CarLifeVehicleInfo refers to car info. For details, please refer to the following Table.

The field definition of CarLifeVehicleInfo in MSG_CMD_CAR_DATA_SUBSCRIBE_DONE is shown in the following table:

Field Name	Type	Optional/ Required	Description
moduleID	int32	Required	The ID of Car Data Modules: <pre>enum ModuleID { CAR_DATA_GPS=0; CAR_DATA_VELOCITY=1; CAR_DATA_GYROSCOPE=2; CAR_DATA_ACCELERATION=3; CAR_DATA_GEAR=4; CAR_DATA_OIL=5; }</pre> among which, CAR_DATA_GPS: The ID of vehicle GPS module; CAR_DATA_VELOCITY: The ID of vehicle speed module; CAR_DATA_GYROSCOPE: The ID of vehicle gyroscope module; CAR_DATA_ACCELERATION: The ID of vehicle acceleration module; CAR_DATA_GEAR: The ID of vehicle gears module; CAR_DATA_OIL: The ID of vehicle fuel module;
supportFlag	int32	Required	0: HU doesn't support the data report of this module. If GPS data is subscribed, change the field description to GPS data format: Two formats are supported now: 1: GPS data format this HU supports is WGS-84, unencrypted format; 2. GPS data format this HU supports is



			GCJ-02, encrypted format; HU can report the GPS data according to the format it supports.
frequency	int32	Required	Car data subscription frequency (Unit: Hz); This frequency can't be lower than the frequency MD requests; If this frequency is lower than the frequency MD requests, MD will not use this car data.

12.9. MSG_CMD_CAR_DATA_SUBSCRIBE_START

The field definition of MSG_CMD_CAR_DATA_SUBSCRIBE_START is shown in the following table:

Field Name	Type	Optional/ Required	Description
vehicleInfo	message CarLifeVehicleInfo	Required	MD requests to subscribe this type of data; HU can start to report this data. Message CarLifeVehicleInfo refers to car data; for details, please refer to the following Table.

The field definition of CarLifeVehicleInfo in MSG_CMD_CAR_DATA_SUBSCRIBE_START is shown in the following table:

Field Name	Type	Optional/ Required	Description
moduleID	int32	Required	The ID of Car Data Modules: enum Module ID { CAR_DATA_GPS=0; CAR_DATA_VELOCITY=1; CAR_DATA_GYROSCOPE=2; CAR_DATA_ACCELERATION=3; CAR_DATA_GEAR=4; CAR_DATA_OIL=5; } among which, CAR_DATA_GPS: The ID of vehicle GPS module; CAR_DATA_VELOCITY: The ID of



			vehicle speed module; CAR_DATA_GYROSCOPE: The ID of vehicle gyroscope module ; CAR_DATA_ACCELERATION: The ID of vehicle acceleration module ; CAR_DATA_GEAR: The ID of vehicle gears module; CAR_DATA_OIL: The ID of vehicle fuel module;
supportFlag	int32	Optional	(Reserved)
frequency	int32	Required	Car data subscription frequency (Unit:Hz);

12.10. MSG_CMD_CAR_DATA_SUBSCRIBE_STOP

The field definition of MSG_CMD_CAR_DATA_SUBSCRIBE_STOP is shown in the following table:

Field Name	Type	Optional/ Required	Description
vehicleInfo	message CarLifeVehicleInfo	Required	MD stops subscribing this type of data; HU shall stop reporting this data. Message CarLifeVehicleInfo refers to car data; for details, please refer to the following Table.

The field definition of CarLifeVehicleInfo in MSG_CMD_CAR_DATA_SUBSCRIBE_STOP is shown in the following table:

Field Name	Type	Optional/ Required	Description
moduleID	int32	Required	The ID of Car Data Modules: enum Module ID { CAR_DATA_GPS=0; CAR_DATA_VELOCITY=1; CAR_DATA_GYROSCOPE=2; CAR_DATA_ACCELERATION=3; CAR_DATA_GEAR=4; CAR_DATA_OIL=5; } among which, CAR_DATA_GPS: The ID of vehicle



			GPS module; CAR_DATA_VELOCITY: The ID of vehicle speed module; CAR_DATA_GYROSCOPE: The ID of vehicle gyroscope module ; CAR_DATA_ACCELERATION: The ID of vehicle acceleration module ; CAR_DATA_GEAR: The ID of vehicle gears module; CAR_DATA_OIL: The ID of vehicle fuel module;
supportFlag	Int32	Optional	(Reserved)
frequency	Int32	Optional	(Reserved)

13. Data Provided to HU from CarLife

To improve user experience in driving, CarLife supports to send the following information to HU, which can display related information on other display areas than HU screen. HU can choose whether to subscribe and respond to the information, among which **Navi TBT info shall follow data subscription mechanism**. For details of **subscription**, please refer to message definition of **MSG_CMD_CARLIFE_DATA_SUBSCRIBE** in **Control Messages**.

13.1. Navigation TBT Info

During CarLife navigation, it can provide the simple guiding information as follows to HU, including the name of the next guiding point, the distance from the next guiding point and the turn type of the next guiding point. Update Strategy shall be in consistent with that of simple guiding information in Navi. **For related communication protocols used to transmit messages, please refer to the following Table.**

The message is sent from MD to HU. For field definition of specific message, please refer to the following Table.

Field Name	Type	Optional/ Required	Description
action	int32	Required	TBT info update type (show, update, hide); for details, please refer to Table 13.1-2.
naviTurn	int32	Required	Type of the next guiding, as an enum value; for details, please



			refer to Table 13.1-3.
roadName	String	Required	Road name of the next guiding point.
totalDistance	int32	Required	Total distance from the next guiding point, in meters.
remaindistance	int32	Required	Remain distance to the next guiding point, in meters
turnIconData	byte	Required	Bitmap binary data for the image of direction indicator

Table 13.1-1 MSG_CMD_NAVI_NEXT_TURN_INFO Message Structure

Enum_Name	Value	Usage
Action_Invalid	0	Invalid value
Action_Show	1	Show
Action_Update	2	Update
Action_Hide	3	Hide

Table 13.1-2 Update Type of TBT Message

Enum_Name	Value	Usage
Turn_Kind_Front	0x00000000 1	Straight
Turn_Kind_Right_Front	0x00000000 2	Turn at the right front
Turn_Kind_Right	0x00000000 3	Turn right
Turn_Kind_Right_Back	0x00000000 4	Turn at the right back
Turn_Kind_Back	0x00000000 5	U-turn
Turn_Kind_Left_Back	0x00000000 6	Turn at the left back
Turn_Kind_Left	0x00000000 7	Turn left
Turn_Kind_Left_Front	0x00000000 8	Turn at the left front
Turn_Kind_Ring	0x00000000 9	Ring
Turn_Kind_RingOut	0x00000000 A	Ring outlet
Turn_Kind_Left_Side	0x00000000 B	Turn left at ordinary/JCT/SAPA two-branch road
Turn_Kind_Right_Side	0x00000000 C	Turn right at ordinary/JCT/SAPA two-branch road



Turn_Kind_Left_Side_Main	0x0000000D	Drive on main line by left side
Turn_Kind_Branch_Left_Main	0x0000000E	Drive on main line by far left side
Turn_Kind_Right_Side_Main	0x0000000F	Drive on main line by right side
Turn_Kind_Branch_Right_Main	0x00000010	Drive on main line by far right side
Turn_Kind_Center_Main	0x00000011	Drive on main line by center
Turn_Kind_Left_Side_IC	0x00000012	Drive on left side of IC two-branch road
Turn_Kind_Right_Side_IC	0x00000013	Drive on right side of IC two-branch road
Turn_Kind_Branch_Left	0x00000014	Drive on the left side of ordinary /JCT/SAPA three-branch road
Turn_Kind_Branch_Right	0x00000015	Drive on the right side of ordinary /JCT/SAPA three-branch road
Turn_Kind_Branch_Center	0x00000016	Drive on the center of ordinary /JCT/SAPA three-branch road
Turn_Kind_Start	0x00000017	Start point
Turn_Kind_Dest	0x00000018	Destination
Turn_Kind_VIA1	0x00000019	Via point 1
Turn_Kind_VIA2	0x0000001A	Via point 2
Turn_Kind_VIA3	0x0000001B	Via point 3
Turn_Kind_VIA4	0x0000001C	Via point 4
Turn_Kind_InFerry	0x0000001D	In ferry
Turn_Kind_OutFerry	0x0000001E	Out ferry
Turn_Kind_TollGate	0x0000001F	Toll gate
Turn_Kind_Left_Side_Straight_IC	0x00000020	Drive on left side of IC two-branch road and go straight
Turn_Kind_Right_Side_Straight_IC	0x00000021	Drive on right side of IC two-branch road and go straight
Turn_Kind_Left_Side_Straight	0x0000002	Drive on left side of



	2	ordinary/JCT/SAPA two-branch road and go straight
Turn_Kind_Right_Side_Straight	0x0000002 3	Drive on right side of ordinary/JCT/SAPA two-branch road and go straight
Turn_Kind_Branch_Left_Straight	0x0000002 4	Drive on the left side of ordinary/JCT/SAPA three-branch road and go straight
Turn_Kind_Branch_Center_Straight	0x0000002 5	Drive on the center of ordinary/JCT/SAPA three-branch road and go straight
Turn_Kind_Branch_Right_Straight	0x0000002 6	Drive on the right side of ordinary/JCT/SAPA three-branch road and go straight
Turn_Kind_Branch_Left_IC	0x0000002 7	Drive on the left side of IC three-branch road
Turn_Kind_Branch_Center_IC	0x0000002 8	Drive on the center IC three-branch road
Turn_Kind_Branch_Right_IC	0x0000002 9	Drive on the right side of IC three-branch road
Turn_Kind_Branch_Left_IC_Straight	0x0000002 A	Drive on the left side of IC three-branch road and go straight
Turn_Kind_Branch_Center_IC_Straight	0x0000002 B	Drive on the center of three-branch road and go straight
Turn_Kind_Branch_Right_IC_Straight	0x0000002 C	Drive on the right side of IC three-branch road and go straight
Turn_Kind_Straight_2Branch_Left_Base	0x0000002 D	Drive straight on the left
Turn_Kind_Straight_2Branch_Right_Base	0x0000002 E	Drive straight on right
Turn_Kind_Straight_3Branch_Left_Base	0x0000002 F	Drive straight on the far left
Turn_Kind_Straight_3Branch_Middle_Base	0x0000003 0	Drive straight on on the center
Turn_Kind_Straight_3Branch_Right_Base	0x0000003 1	Drive straight on the far right
Turn_Kind_Left_2Branch_Left_Base	0x0000003 2	Turn left and then drive on the left
Turn_Kind_Left_2Branch_Right_Base	0x0000003 3	Turn left and then drive on the right
Turn_Kind_Left_3Branch_Left_Base	0x0000003 4	Turn left and then drive on the far left
Turn_Kind_Left_3Branch_Middle_Base	0x0000003	Turn left and then drive on the



	5	center
Turn_Kind_Left_3Branch_Right_Base	0x00000036	Turn left and then drive on the far right
Turn_Kind_Right_2Branch_Left_Base	0x00000037	Turn right and then drive on the left
Turn_Kind_Right_2Branch_Right_Base	0x00000038	Turn right and then drive on the right
Turn_Kind_Right_3Branch_Left_Base	0x00000039	Turn right and then drive on the far left
Turn_Kind_Right_3Branch_Middle_Base	0x0000003A	Turn right and then drive on the center
Turn_Kind_Right_3Branch_Right_Base	0x0000003B	Turn right and then drive on the far right
Turn_Kind_Left_Front_2Branch_Left_Base	0x0000003C	Drive on the left front and then to the left side
Turn_Kind_Left_Front_2Branch_Right_Base	0x0000003D	Drive on the left front and then to the right side
Turn_Kind_Right_Front_2Branch_Left_Base	0x0000003E	Drive on the right front and then to the left side
Turn_Kind_Right_Front_2Branch_Right_Base	0x0000003F	Drive on the right front and then to the right side
Turn_Kind_Back_2Branch_Left_Base	0x00000040	U-turn and then drive on the left
Turn_Kind_Back_2Branch_Right_Base	0x00000041	U-turn and then drive on the right
Turn_Kind_Back_3Branch_Left_Base	0x00000042	U-turn and then drive on the far left
Turn_Kind_Back_3Branch_Middle_Base	0x00000043	U-turn and then drive on the center
Turn_Kind_Back_3Branch_Right_Base	0x00000044	U-turn and then drive on the far right
Turn_Kind_Ring_Front	0x00000045	Round ring and drive front
Turn_Kind_Ring_Left	0x00000046	Round ring and turn left
Turn_Kind_Ring_LeftBack	0x00000047	Round ring and turn leftback
Turn_Kind_Ring_LeftFront	0x00000048	Round ring and turn leftfront
Turn_Kind_Ring_Right	0x00000049	Round ring and turn right
Turn_Kind_Ring_RightBack	0x0000004A	Round ring and turn rightback



Turn_Kind_Ring_RingtFront	0x0000004 B	Round ring and turn rightfront
Turn_Kind_Ring_TurnBack	0x0000004 C	Round ring and turn back

Table 13.1-3 Enumerated values of Guiding Type

For the corresponding reference figure of guiding types, please refer to the document *Description of CarLife Navi HUD Turn Type Sent from MD to HU.docx*.

In navigation process or e-dog process, CarLife can offer the following Assistant Guiding Info from HU, including: Assistant Guiding Mark Type (such as speed camera), distance to the next guiding point, and Guiding Table, Picture & Data. Update Strategy shall be in consistent with that of assistant guiding information in Navi and e-dog. For details, please refer to the Message Structure Definition in the following Table:

Field Name	Type	Optional/ Required	Description
action	int32	Required	TBT info update type (show, update, hide); for details, please refer to Table 15-2.
assistantType	int32	Required	Type of Assistant Guiding Info, as an enum value; for details, please refer to Table 15-5.
trafficSignType	int32	Required	Subtype of Traffic Guiding Info, as an enum value; this field is valid when assistantType is trafficSign. Reserve the field; not used yet; default value as 0.
totalDistance	int32	Required	Total distance to the next assistant guiding point, in meters
remainDistance	int32	Required	Remain distance to the next assistant guiding point, in meters
cameraSpeed	int32	Required	Speed limits of speed camera, in m/s. This field is only valid when assistantType is Assistant_Type_SpeedCamera.

Table 13.1-4 MSG_CMD_NAVI_AssitantGuide_INFO Message Structure

Enum_Name	Value	Usage
Assistant_Type_Invalid	0	Invalid value
Assistant_Type_trafficSign	1	Other traffic signs than camera (reserved field, not used yet)
Assistant_Type_SpeedCamera	2	Speed camera



Assistant_Type_TrafficLightCamera	3	Traffic signals camera
Assistant_Type_PeccanryCamera	4	Peccanry camera
Assistant_Type_IntervalCamera	5	Interval camera

Table 13.1-5 AssistantType Enumerated Type Definition

13.2. Music Play Info

When CarLife plays music, it can provide the name, singer and length of the song being played to HU. Update Strategy is to refresh when starting playing or cutting songs. For details, please refer to the following Table:

Field Name	Type	Optional/ Required	Description
source	string	Required	Source of the song being played
song	string	Required	Name of the song being played
artist	string	Required	Singer (Artist) of the song being played
album	string	Required	Album name of the song being played
albumArt	byte	Required	Bitmap binary data of cover picture of the song being played
duration	int32	Required	Length of the song being played, in seconds.
playlistNum	int32	Required	Number of songs in the current play list
songId	string	Required	ID of the song being played (For CarLife's internal use)
mode	int32	Required	Play mode as an enum. For more details please refer to Table 13.2-2

Table 13.2-1 MSG_CMD_MEDIA_INFO Structure Definition

Enum_Name	Value	Usage
PLAYMODE_REPEAT_ONE	0	Repeat One
PLAYMODE_SHUFFLE	1	Shuffle
PLAYMODE_REPEAT_ALL	2	Repeat All

Table 13.2-2 PlayMode Enumerated Definition



13.3. Music Play Progress Info

When CarLife plays music, it can provide the progress info of the song being played to HU. Update Strategy is to refresh per second. For details, please refer to the following Table:

Field Name	Type	Optional/ Required	Description
progressBar	Int32	Required	Progress info of the song being played (in m/s)

Table 13.3-1: MSG_CMD_MEDIA_PROGRESS_BAR Structure Definition

14. Exiting Mechanism of CarLife on HU

Under connected status, users can click the 'CarLife' button on CarLife Homepage to enter a sub-page; when users click 'go back to HU' button on the sub-page, MD will send MSG_CMD_GO_TO_DESKTOP message to HU, which can choose between the following 2 responses according to its own requirements:

- Completely quit CarLife on HU
- Switch CarLife to HU Desktop

15. Update Strategy

15.1. CarLife on HU

As a HU side solution which highly conforms to vehicle standards, CarLife can keep its HU features stable for a long time; normally no update is needed on HU; update of all functions takes up on MD. However, in order to protect the comprehensiveness of CarLife system, we have regulated HU update strategy which is compatible with OEMs' habits. OEMs can choose between the following two update strategies according to their own system:

- Update CarLife in 4S Shops
- OTA Update

In case of CarLife on HU has critical updates, CarLife team will inform OEM partners ASAP. OEMs can decide whether to update CarLife on HU or not.

Before OEMs update CarLife on HU, acceptance process from CarLife team must be finished.



15.2. CarLife on MD

CarLife on MD will regularly update the version in mainstream Android App markets and Apple's App Store according to its internal version plan. The update process here is no different from that of other APPs on MD.

15.3. Compatibility Strategy

15.3.1. CarLife on HU

CarLife on HU has a very light structure with nearly zero product logistics. Basic functions like connection, audio & video stream and back control will guarantee the applicability of CarLife. Subsequent update will only add new features rather than revise the original features. Therefore, as long as OEMs follow CarLife Protocols and Integration Specifications, different versions on HU are compatible.

15.3.2. CarLife on MD

Any Update of CarLife on MD won't affect the version of CarLife on HU.

15.3.3. Data Protocol

CarLife on HU & MD have related protocol version number to solve compatibility issues between CarLife on HU & MD.

Version number's rule has two digits numbers, first digit number means the main version and second digit number means the minor version. Currently in CarLife, the protocol version number of HU and MD has been initialized as 1.0. The rules of protocol version number as shown below:

- When HU and MD use the same protocol version number, then HU and MD's data protocols are compatible.
- If old data protocol and new data protocol are not compatible, such as modifying original data protocol's pb structure, then HU and MD's main version number should add one digit number and minor version number should add nothing.
- Normally, new data protocol version will only add new protocols, and won't modify the



original protocol. Therefore, there won't be any incompatible issues regarding to old version and new version data protocol. When new protocol adds new messages which send from MD to HU (only sending relevant data to HU won't affect the normal working of current functions), HU and MD's minor version number should add one digit number, and main version number won't be affected.

16. Safety Verification

16.1. Safety Verification of CarLife on MD to CarLife on HU

On one hand, CarLife on MD uses the Safety Verification process to verify whether the connecting HU conforms to CarLife Standard. On the other hand, CarLife on MD also uses the authentication process to identify the specific type of the connecting HU, in order to provide customized services and content.

CarLife Safety Verification is an optional function. HU OEMs can decide whether to support according their own conditions.

16.1.1. Definition of Verification Message

CarLife on MD verifies CarLife on HU via MSG_CMD_STATISTIC_INFO. For specific definition of this message, please refer to Chapter 10.2.2.

The field definition of MSG_CMD_MD_AUTHEN_RESULT is shown in the following Table 16.1-1:

Table 16.1-1 Field Definition of MSG_CMD_MD_AUTHEN_RESULT

Field Name	Type	Optional/ Required	Description
result	bool	Required	The authentication result of CarLife on MD verifying CarLife on HU

16.1.2 Verification Process

The verification process of CarLife on MD to CarLife on HU is shown in Figure 16.1-1:

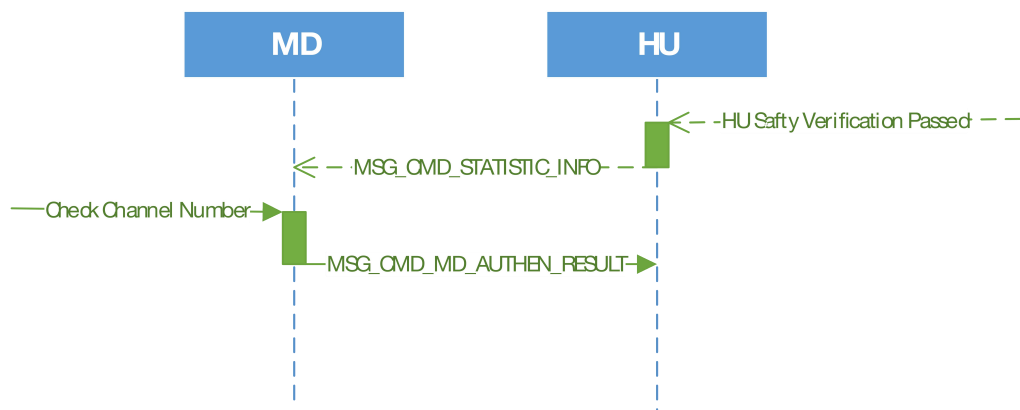


Figure 16.1-1: The Verification Process of CarLife on MD to CarLife on HU

1. After CarLife on MD is verified by CarLife on HU, HU will send statistic information via MSG_CMD_STATISTIC_INFO to MD.
2. When CarLife on MD receives the statistic information, it will judge whether this HU is legal HU according to its Channel Number.
3. If the HU is legal, the verification process of CarLife on MD to CarLife on HU will be finished. Meanwhile the verification result will be sent to HU via MSG_CMD_MD_AUTHEN_RESULT from MD, activating the corresponding customized service based on OEMs;
4. If the HU is illegal, the verification result will be sent to HU via MSG_CMD_MD_AUTHEN_RESULT from MD and MD will automatically disconnect the connection.

17. Performance Indicators

Performance Indicators indicate the overall performance and user experience related indicators when HU runs CarLife. For details, please refer to Table 17-1:

Table 17-1

No.	Indicator	Description
#1	CPU	HU CPU status can guarantee HU is using CPU correctly and reasonably while running CarLife on HU. It focuses on CPU usage percentage while running CarLife. Recommended percentage is lower than 20%.
#2	RAM	The RAM standard of CarLife on HU focuses on distinguish RAM usage when CarLife is running on different module, such as Navigation, Music, Navigation+Music etc. Recommended number is lower than 50M.
#3	Video stream	Frame rate guarantees the fluency when users connect and use CarLife. Recommended number is 30 frames/s, without obvious lag,



	Frame rate	latency and losing frame.
#4	Latency	Latency guarantees the synchronized performance when video stream transmitting. Recommended number is lower than 200 ms.
#5	Microphone on HU	Speech recognition rate used to measure the vehicle computer MIC meets CarLife requirements, the use of the vehicle-side recording software to record and analyze their corpus depot itself no noise (noise does not affect voice) or truncation. For details, please refer to 8.7 VR Function
#6	Connection time	Connection time can measure whether the time used to establish connection between MD and HU can meet CarLife requirement. Recommended number is lower than 4 seconds.

18. Staticstic Info Clarification

In order to provide better services, CarLife needs to transmit related info on HU to MD via the protocols between HU and MD, for MD to report the statistic info. There are 2 related messages in the protocols:

- MSG_CMD_HU_INFO: HU system info
- MSG_CMD_STATISTIC_INFO: channel and running info

The following paragraphs will introduce the definition and use of fields in the 2 messages.

18.1. CarLife on HU Statistic Info Clarification (MSG_CMD_HU_INFO)

CarLife on HU Statistic Info (namely, MSG_CMD_HU_INFO) contains basic hardware and software info of HU system, which can be used to track and locate problems, help us find the cause of problems and solve them faster. If online BUG appears, we will locate the malfunctioned HU quickly via the reported HU system info, restore the environment of the BUG (hardware & software environment), locate and solve the BUG quickly.

This message contains the following fields, which correspond to different system info:

Field Name	Type	Optional/Required	Description
os	String	Optional	HU OS, includes: QNX, Linux, WinCE and Android.
manufacturer	String	Optional	HU manufacturer's name, such as Hyundai
model	String	Optional	HU model number, such as:



			HD_MODEL_112
release	String	Optional	Release version name of HU system, such as HDCS1.0

Table 18.1-1

Fields above are all optional. However, more information can help CarLife team to follow and locate problems. Once CarLife has been successfully connected, HU can send above info to MD, which will save the info and report it for analysis when exception occurs.

18.2. Channel and Running Statistic Info Clarification (MSG_CMD_STATISTIC_INFO)

CarLife on HU will use message 'MSG_CMD_STATISTIC_INFO' to gather HU channel information and running information. These info can help CarLife team to know the running condition and log of CarLife on HU. This can help CarLife improve stability, capability and optimize user experience.

This message includes the following fields, which stand for different meanings:

Field Name	Type	Optional/Required	Description
cuid	String	Required	A unique ID for each HU, used for identifying each HU.
versionName	String	Required	Version name of CarLife on HU, such as 1.0.0, 1.1.0
versionCode	Integer	Required	Version number of CarLife on HU, such as 1, 2
channel	String	Required	Channel number of CarLife on HU, assigned by Baidu and used for uniquely identifying each brand of HU.
connectCount	Integer	Required	It means times for setting up connection between HU and MD. Once successfully connected, this parameter should be cleared (zero). This is used for calculating the rate of successful connection.
connectSuccessCount	Integer	Required	Times of successful connection, normally set as 1, used for calculating successful connection rate.
connectTime	Integer	Required	Time spent for connection, used for figuring out the average time consuming for connection.



crashLog	String	Required	Critical log of last crash on HU, used for locating issues and follow up.
----------	--------	----------	---

Table18.2-1

When HU and MD are successful connected, HU must send this message to MD within 2 minutes (for details, please refer to [Sequence Diagram](#)). Otherwise, MD will disconnect with HU. When MD receives this message, it will send info above to server for recording running status and crash log.

19. Support from CarLife Team to HU OEMs

19.1 Key Test Tools

19.1.1 CarLife Protocol Tools

CarLife Protocol Tools are mainly used to test and output info interaction, MD Switch to Background, Hard Keys, Audio as well as other real-time interactive info when HU and MD are connecting, so as to locate problems and verify info during the development process.

Protocol message analysis and response steps when HU and Andriod MD are connecting:

1. Log on <http://180.76.233.125:8080/login.jsp> with account and password assigned by Baidu, and add IMEI number to test Andriod MD;
2. Install Self-test installation package offered by Baidu on the test MD (HU Protocol message report function and analog Navi function have been activated on the MD);
3. Keep MD's internet connected, launch CarLife App on MD and start function test; select test devices on Web, on which real-time protocol message will be displayed;

Protocol message analysis and response steps when HU and iOS MD are connecting:

1. Install RemoteLogger tool on PC (MAC), make sure that PC and test MD use the same WIFI (which supports broadcast);
2. Install Self-test installation package offered by Baidu on the test MD (HU Protocol message report function and analog Navi function have been activated on the MD);
3. After test MD is successfully connected with RemoteLogger tool, connect test MD with HU to have function test;
4. RemoteLogger tool on PC will display real-time protocol message log.

20. CarLife Certification

In order to better guarantee the quality and user experience of CarLife HU & MD Integration Products, Baidu formulates this certification process. If certification fails, partners shall modify and apply for certification again until CarLife Certification passes; only then can partners start mass production.

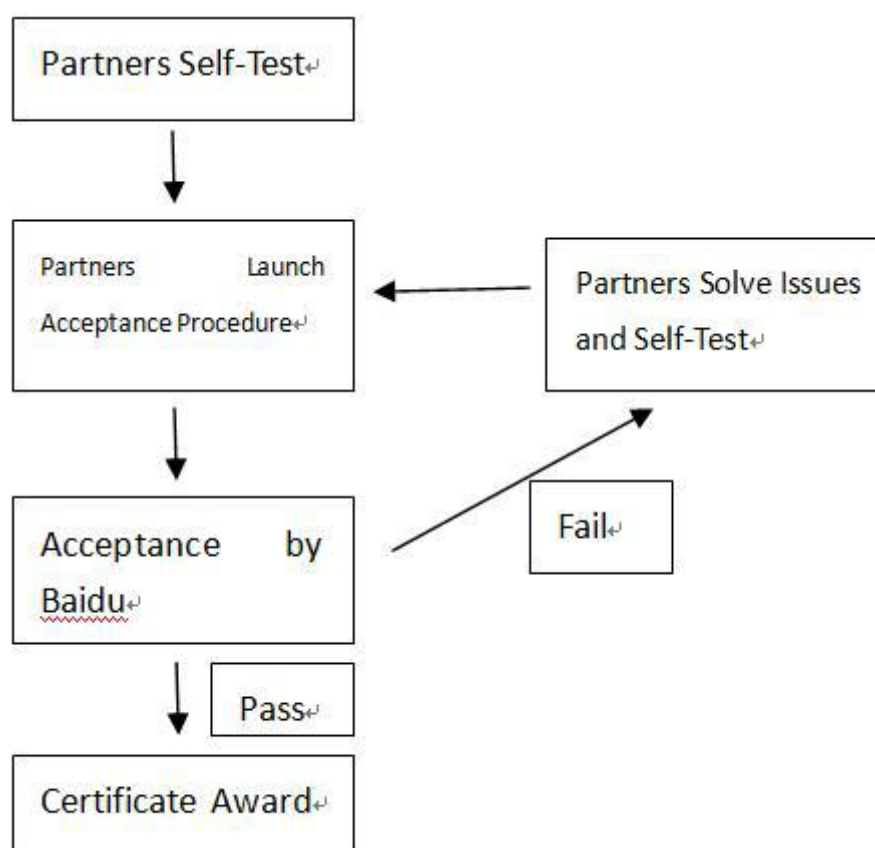


Figure 20-1: CarLife Certification Process

20.1. Partners Self-Test and Launch Acceptance Process

Quality of versions is the responsibility of partners, who shall conduct system test when the development process is finished. For related questions, please follow up [CarLife HU & MD Integration FAQ.xls](#); for question modules, please fill in [CarLife HU & MD Integration Question Modules.xls](#).

After partners have conducted a full self-test:

1. Log on <http://carlife.baidu.com/> - Log on Baidu account – Click “I am OEM/Tier 1” – Enter



“My Project” to choose the projects which require certification, fill in required info. Correct CH & EN car model name and deliver HU tracking number and project self-test report shall be provided. (Partners must have project established in “I am OEM/Tier 1” of CarLife website)

2. Partners can also apply for acceptance to Baidu (send e-mails to carlife@baidu.com)

Required materials for certification are:

1. CarLife HU Software and Hardware Environment

a) HU system which installs the latest version of HU CarLife and related accessories to use CarLife (such as microphone, power, speaker);

Note: If HU system or CarLife version has update, please notify CarLife team in time.

2. Related Test Reports & Documents

For details, please refer to Table 20.1-1:

No.	Name	Type	Medium
1	Launching CarLife Instruction (on HU)	Documents	e-edition/Paper
2	System Parameters Configuration Instruction	Documents	e-edition/Paper
3	System Function Test Report	xls	e-edition/Paper
4	System Performance Test Report	xls	e-edition/Paper
5	Stability Test Report	xls	e-edition/Paper
6	Compatibility Test Report	xls	e-edition/Paper
7	Issues Record Form	xls	e-edition/Paper

- 1) Launching CarLife Instruction (on HU)——Instruction of how users connect partners' HU with MD;
- 2) System Parameters Configuration Instruction——Instruction of channel number, protocol version number and statistic information
- 3) System Function Test Report——Test report to clarify CarLife on HU passed the basic function test, for details please refer to *CarLife on HU Basic Functions Case.xls*
- 4) System Performance Test Report——for details please refer to 'Performance Indicators' Chapter.
- 5) Stability Test Report——CarLife team recommends to continuously run CarLife for 4 hours * three times (HU has no disconnection or crash; recommend road test), when running navigation and playing music at the same time.
- 6) Compatibility Test Report——Such testing features included but not limited to different Smartphone OS, smartphone brands, OS version, screen resolution and different versions of CarLife on MD
- 7) Issues Record Form——HU partners should cooperate with Baidu in terms of composing this document, during development, test, certification and regression testing. All issues found in above procedures should be recorded in this document.



For details, please refer to *CarLife HU & MD Integration Questions Modules.xls*

20.2. Certification Acceptance by Baidu

After partners apply for acceptance process, Baidu will conduct a series of full-function acceptance tests according to documents and reports provided by Baidu. After the tests are finished, Baidu will provide corresponding test reports and question list. Baidu will decide whether the acceptance is successful according to *HU CarLife Acceptance Success Standards*.

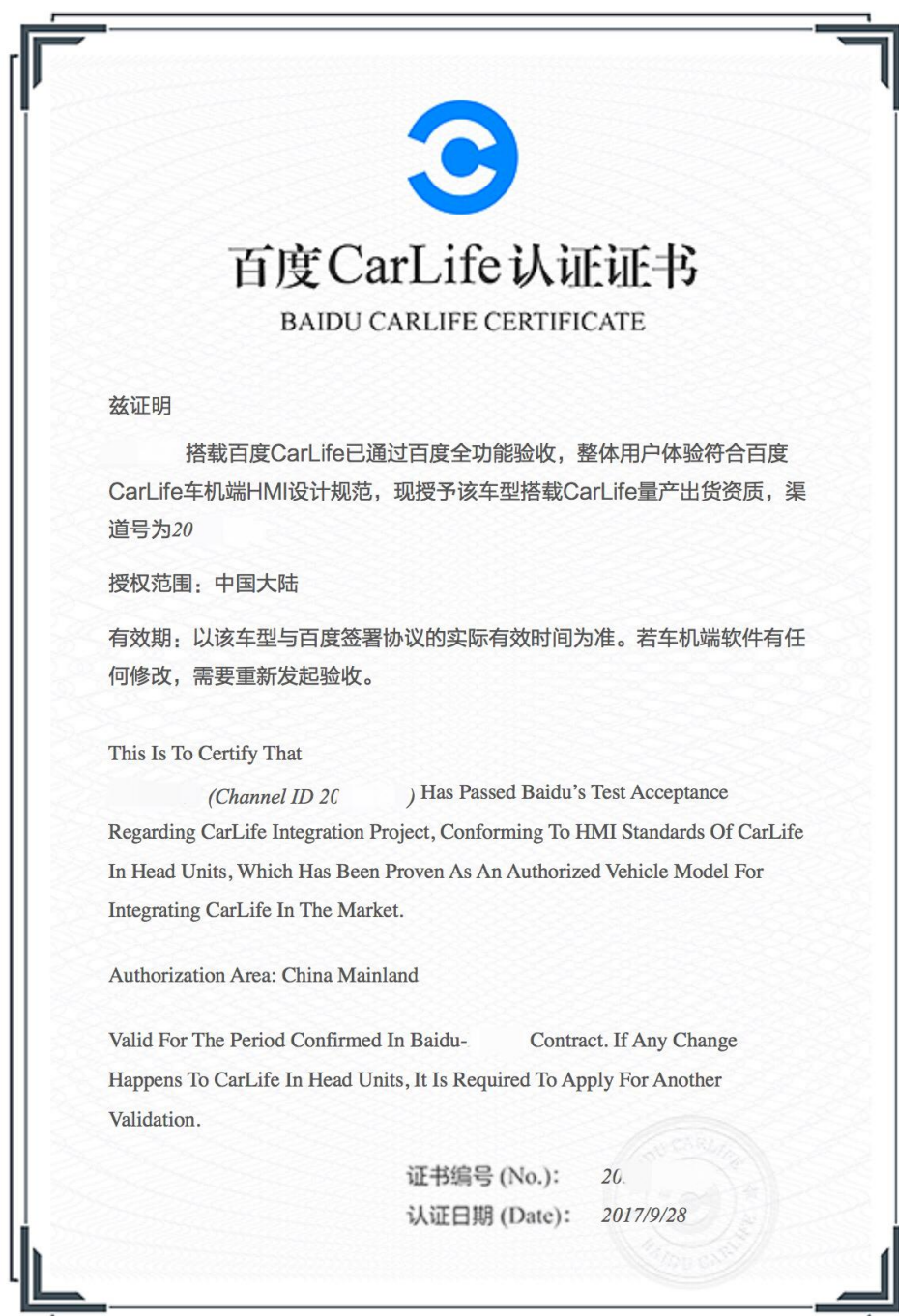
During the acceptance process, Baidu will conduct the following three steps which will take 1-2 weeks:

1. Acceptance of reports provided by partners, including function test, performance test, compatibility test, stability test, etc.
2. Test HU, including video, audio, connection, **VR, back control**, etc.
3. Record issues found in the acceptance, fill in the Issues Record Form and synchronize the results to partners to solve the issues together.

Mechanism of solving issues found in testing:

- a) If HU causes that users cannot normally use, it must be solved;
- b) For issues which rarely occur and will not influence normal use, it is recommended to lower down priority and follow up later;
- c) MD issues will not be part of OEM partner acceptance standard; MD issues can be solved when MD version updates.

After successful acceptance, Baidu will award certificate for mass production; certificate sample is listed as follows:



21. HU HMI Design Standards

In order to help our partners to better understand how CarLife Integration should be presented ultimately on HU, we briefly summarize related HMI design interfaces and interactive contents for reference, based on previous development and cooperation. For specific details, please



refer to the full version of *CarLife HMI Design Standards for Head Unit*.

21.1. The Standard for CarLife Icon and App Title on HU

Head Unit: required to set a CarLife entry point in the menu. In this case, we offer the CarLife Icon (500x500) standard documents to allow partners to adjust the size, insuring it match with their Head Unit HMI design. Meanwhile we also provide the icon for CarLife navigation function individually as a shortcut.

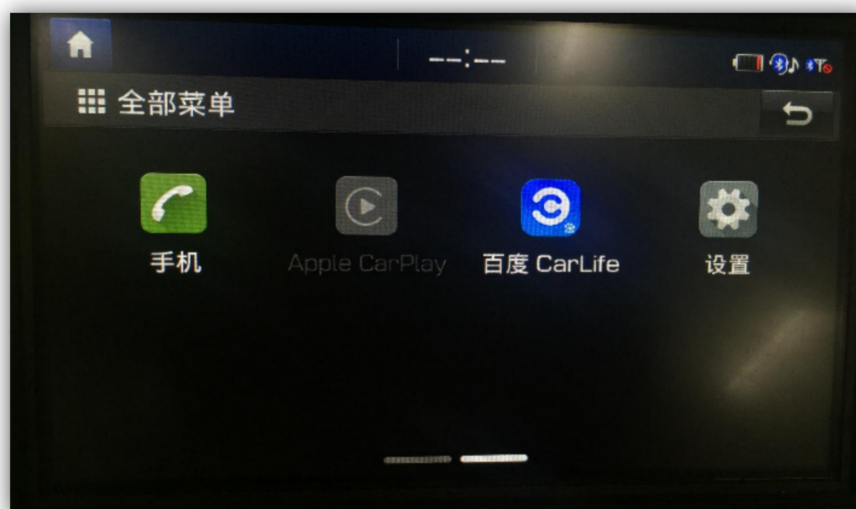
Figure 21.1-1 CarLife Icon Standard Documents



Figure 21.1-2 CarLife Navigation Shortcut Icon



Figure 21.1-3 The Sample of CarLife Icon on HU Menu





The title language (CH&EN) of CarLife will change according to the system language of HU, please follow the patterns:

- Chinese title: 百度 CarLife
- English title: Baidu CarLife

**Please notice that the name of Baidu and CarLife is case sensitive*

21.2. The Standard of Connection Related Interaction on Head Unit

HU requires to inform users when non-video stream interfaces are shown, regarding to different connection related cases in the process of using CarLife. The rest of this chapter will introduce how Head Unit should guide users during these scenarios. Partners could choose to either adopt CarLife official design directly or self-design basing on scenarios.

**The following figures are interaction sketches for reference only.*

21.2.1. Before Connection

If users click the CarLife Icon on Head Unit menu without the connection, Head Unit should prompt users to connect.

Figure 21.2-1 Sketch for HU Prompting No Connection



21.2.2. No Installation of CarLife on MD

When HU detects there is no installation of CarLife on MD after successful connection, it should show a QR code for downloading CarLife and a button for re-connection.

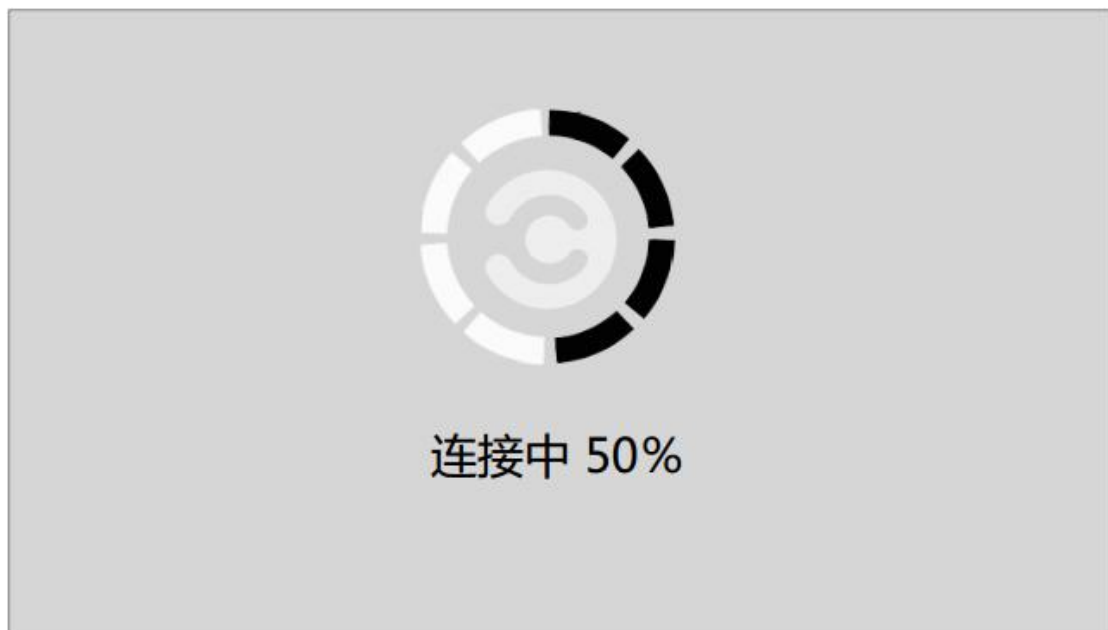
Figure 21.2-2 Sketch for HU Prompting to Download CarLife (QR Code)



21.2.3. Connecting

HU should show loading status after connection is built up as follows:

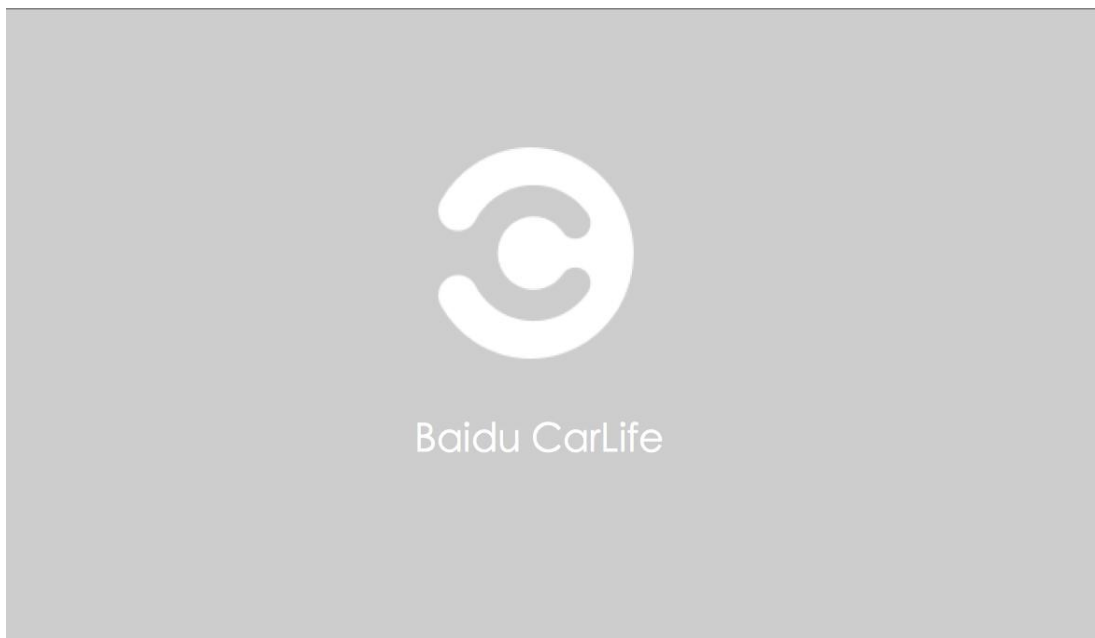
Figure 21.2-3 Sketch for HU Loading the CarLife Connection



21.2.4. Connected Successfully

After connection is successfully finished, HU will show a bit of animation for confirming connection has been built up before entering CarLife launch page and homepage.

Figure 21.2-4 Sketch for The Animation of Successful Connection on HU



21.2.5. Connection Failed/Disconnected

If users manually terminate the connection or the connection is detected failed for other reasons, HU will prompt the users to retry the connection or check the connection help.

Figure 21.2-5 Sketch for Connection Failed/Break on HU



21.2.6. Connection Help

Users could view the connection help through  icon at the top right corner of the connection failed interface on Head Unit.

Figure 21.2-6 Sketch for Connection Help on HU



21.2.7. CarLife in Background/Screen Locked

If users manually switch CarLife App to background on MD or lock the screen, HU will inform users that CarLife is not available in this situation, please unlock the screen or bring CarLife back to foreground on MD. A re-launched button for bring the APP to foreground is preferred to show on Head Unit as well.

Figure 21.2-7 Sketch for Background/Phone Locked status on HU

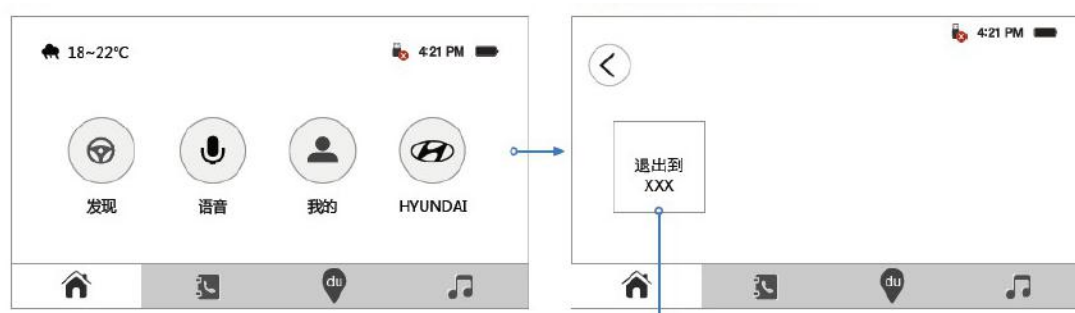


21.2.8. Exit/Return CarLife

After connecting successfully, the CarLife Button in the rightmost of homepage will change to a button with corresponding OEMs logo. Users could switch between the native Head Unit system and CarLife interactive screen via the first exit/back button in a subpage after clicking this OEMs button. CarLife offers two intentions for this exit/back button in OEMs button's subpage:

- Back to HU native system.
- Exit CarLife on Head Unit completely.

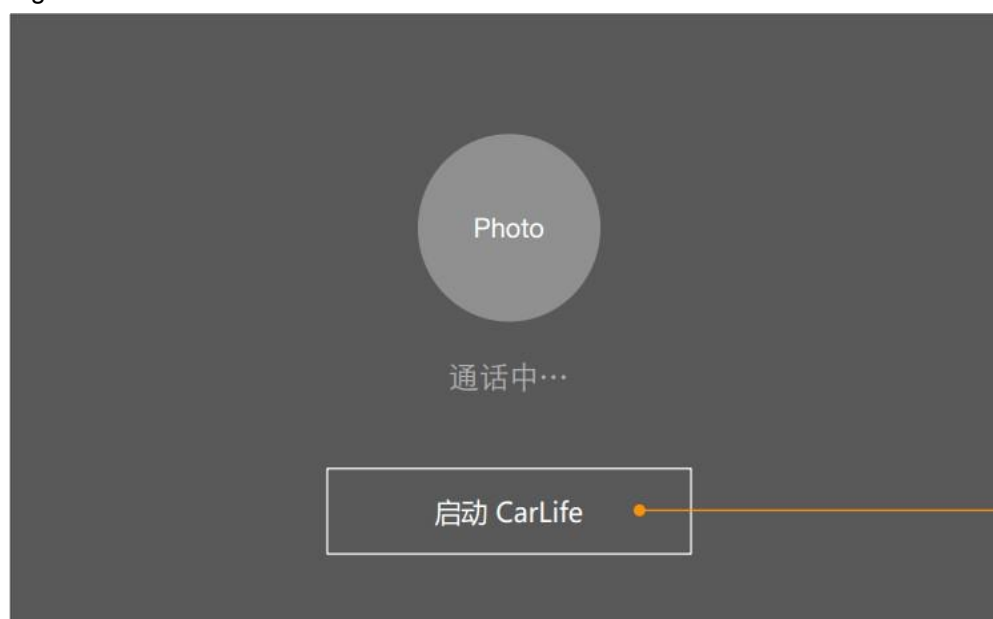
Figure 21.2-8 Sketch for CarLife Button/OEMs Button



21.2.9. Bluetooth NOT Connected

If HU BT and MD BT are not connected, MD shall display CarLife *On the Phone* prompt page no matter whether CarLife BT Phone Call function has been adapted.

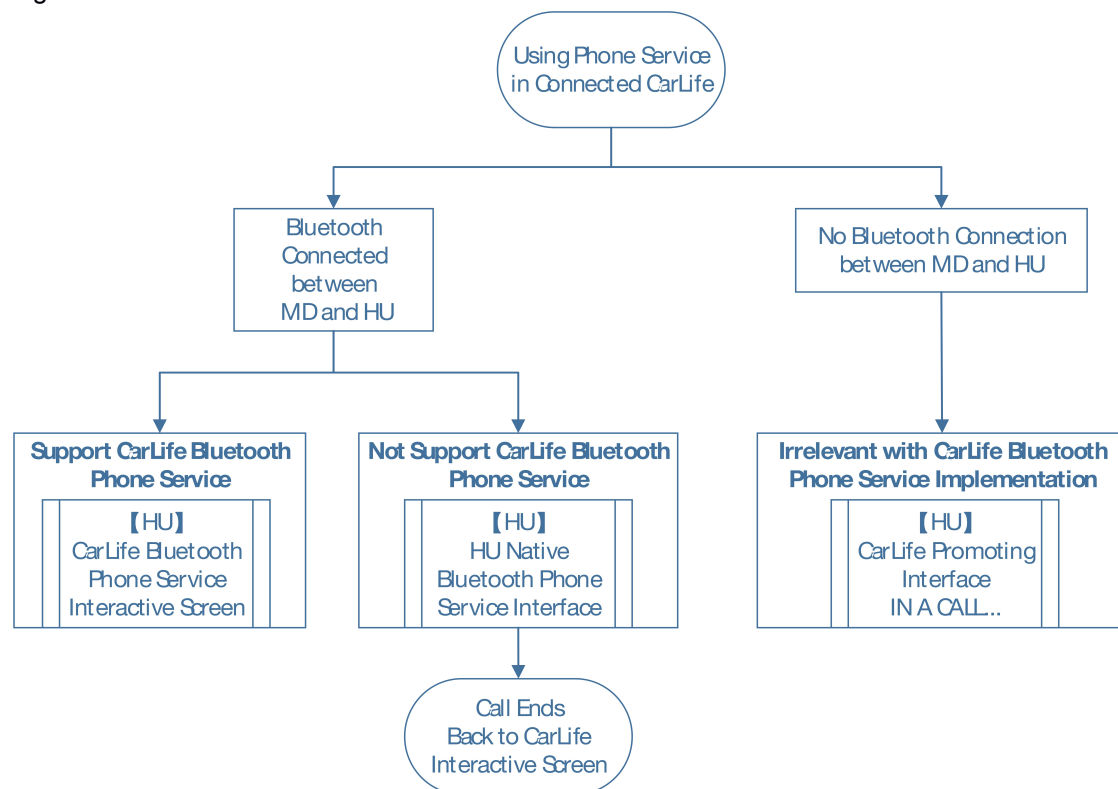
Figure 21.2-9: Sketch for CarLife *On the Phone* for HU



21.3. Phone Related Interaction

After MD CarLife is successfully connected with HU CarLife, using connected CarLife to make or receive calls involves a series of interaction with Head Unit. In order to achieve making or answering phone calls in CarLife Internal UI, it requires to satisfy two conditions: not only Mobile Devices need to connect with Head Units Bluetooth but also HU should implement CarLife Bluetooth phone function.

Figure 21.3-1 The Interaction of Phone Service in Connected CarLife with HU



21.3.1. Bluetooth Connected

- **Not Implement CarLife Bluetooth Phone Service**

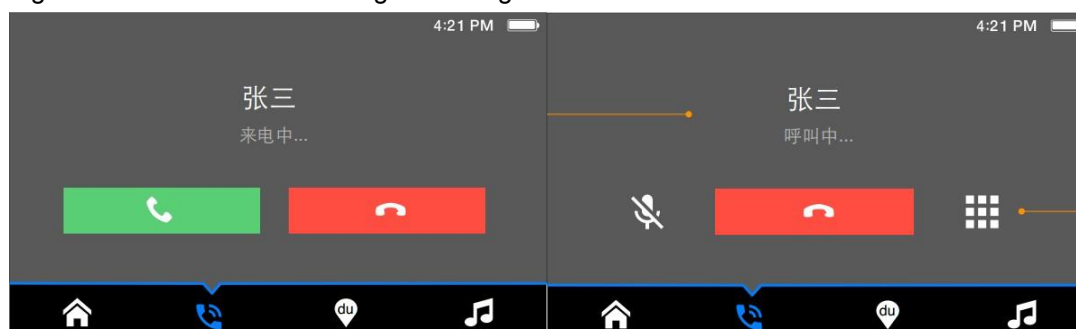
When Bluetooth connected while no implementation of CarLife Bluetooth phone service on Head Unit, HU should switch to native Bluetooth phone interface for making or receiving phone calls.

- **Implement CarLife Bluetooth Phone Service**

When Bluetooth connected and CarLife Bluetooth phone service also implemented, HU should stay in the internal CarLife UI for making or receiving phone calls.



Figure 21.3-2 Sketch for Making/Receiving Calls with CarLife Internal UI on Head Unit



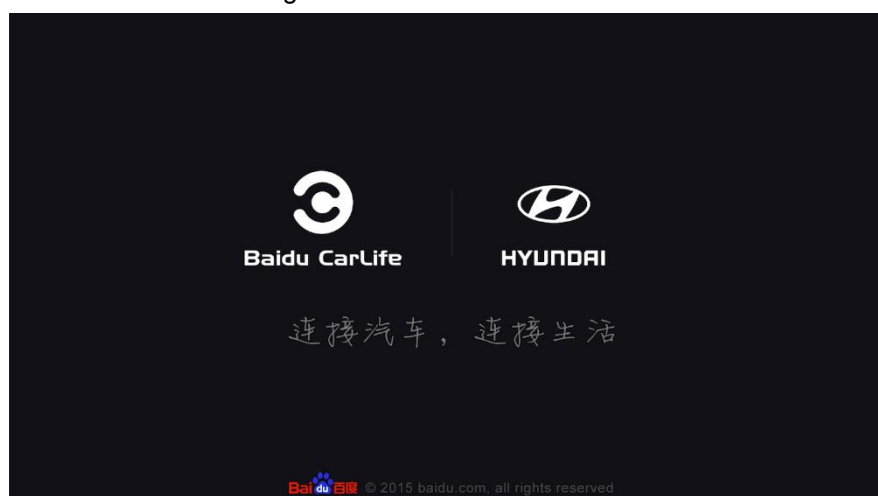
21.4. The Standard of Partners' Brand Customization on HU

CarLife offers two options for partners' brand exposure in UI design of CarLife on MD: OEMs button on launch page and APP Homepage. The customization requires partners to provide source file (AI or PSD) of brand logo. The final design will be confirmed with partners before implementation.

21.4.1. CarLife Launch Page on HU

Once CarLife icon is clicked on HU, the screen will show the launch page before connection initialization. For this brand customized launch page, it will contain both CarLife logo (left side) and car brand logo (right side). In this case, white version of the logo is preferred in order to keep the consistency.

Figure 21.4-1 CarLife Launch Page with HYUNDAI



21.4.2. OEMs Button

When MD detects the car brand after successful connection, the 4th round button on homepage will automatically switch to car brand button. White version of brand logo and English brand name will be set as default.

Figure 21.4-2 CarLife Homepage of Mercedes



21.5. Activating CarLife by Default

CarLife is required to activate by default when co-existing with other smartphone integrated applications on Head Unit. The native Head Unit settings should allow users to choose the default startup.