



CarLife 车机手机互联方案接入详细指南

目录

CARLIFE 车机手机互联方案接入详细指南	1
1. 版本历史	7
2. 前言	10
2.1. 背景	10
2.2. CARLIFE 车机端整体架构	10
2.2.1. 传输介质	10
2.2.2. CarLife 方案整体框架	11
2.2.3. 车机端架构	11
2.3. 通信数据的载体-PB	12
2.4. 名词解释	13
3. 设备要求	13
3.1. 车机端最低硬件要求	14
3.2. 手机端要求	15
4. 车机端与 CARLIFE 关系密切的能力	15
5. 数据通道	15
6. 连接引擎	16
6.1. 连接方式	16
6.1.1. AOA 连接	16
6.1.1.1. AOA 简介	16
6.1.1.1.1. Android USB 通信模式	17
6.1.1.1.2. Android 配件、ADK 以及 AOA 协议	18
6.1.1.2. AOA 设备握手	18
6.1.1.2.1. AOA 设备握手过程	18
6.1.1.2.2. AOA 设备握手指令	22
6.1.1.2.3. AOA 设备握手流程图	23
6.1.1.3. AOA 消息结构	23
6.1.1.4. AOA 车机端开发	24
6.1.1.5. AOA 数据通道建立	25
6.1.1.6. 车机端 CarLife AOA 连接动画百分比定义	25
6.1.1.7. 移动设备端 CarLife 应用拉起与判别	26
6.1.1.7.1. 移动设备端 CarLife 拉起	26
6.1.1.7.2. 移动设备端是否安装 CarLife 应用	26
6.1.2. EAP 连接	26
6.1.2.1. 车机端的开发工作	27
4、主从模式切换	27
6.1.2.2. USB 连接层开发	28
6.1.2.2.1. 连接过程	28
6.1.2.2.2. 模式切换	31

6.1.2.2.3.	上报的 USB 端口	33
6.1.2.3.	鉴权	34
6.1.2.3.1.	Accessory Authentication	34
6.1.2.3.2.	Accessory Identification	36
6.1.2.4.	拉起手机端 CarLife	37
6.1.2.5.	EAP 连接消息结构	38
6.1.2.6.	车机驱动层和车机端 CarLife 的交互协议	39
6.1.2.6.1.	驱动和车机端 CarLife 的通信方式	39
6.1.2.6.2.	车机驱动发送给车机端 CarLife 的消息	39
6.1.2.6.3.	车机端 CarLife 发送给驱动的消息	39
6.1.2.7.	车机端 CarLife 增加对手机的心跳消息	40
6.1.2.8.	车机端 CarLife 从 EA 通道读取数据的方法	40
6.1.2.9.	车机端 CarLife EAP 连接动画百分比定义	40
6.1.2.10.	CarLife 与 CarPlay 的区别	41
6.1.2.11.	CarLife 与 CarPlay 在车机端共存的拉起策略	41
6.1.2.12.	CarLife MFi 认证流程	43
6.2.	蓝牙自动匹配（车机端可选择性支持）	46
6.2.1.	蓝牙配对模式	47
6.2.1.1.	PIN CODE 模式	47
6.2.1.2.	SSP 模式	47
6.2.2.	自动匹配流程	47
6.2.3.	初始化自动匹配内置蓝牙电话功能	49
6.2.4.	自动匹配接口定义	50
6.2.4.1.	MSG_CMD_HU_BT_OOB_INFO	51
6.2.4.2.	MSG_CMD_MD_BT_OOB_INFO	51
6.2.4.3.	MSG_CMD_START_BT_AUTOPAIR	52
7.	视频引擎	52
7.1.	BDSC 的处理方式	52
7.2.	动态帧率机制	53
7.2.1.	车机侧的帧率调整	53
7.2.2.	移动设备侧的帧率调整	53
7.3.	动态分辨率机制	53
7.4.	超宽或者超高屏幕适配策略	53
8.	音频引擎	54
8.1.	CARLIFE 音频引擎架构	54
8.2.	音频流类型	55
8.3.	音频焦点类型	56
8.4.	音频焦点管理策略	57
8.4.1.	duck 处理	58
8.4.2.	mute 机制	59
8.5.	与车机本地音频的冲突处理策略	60
8.6.	音频应用优先级	60
8.7.	VR 功能	60

9. 蓝牙电话	62
9.1. 前言	62
9.2. 接口框架	63
9.2.1. 整体结构	63
9.2.2. 蓝牙电话消息序列	64
9.3. 接口定义	65
9.3.1. 蓝牙电话主要功能	65
9.3.2. 具体接口定义	65
9.3.2.1. MSG_CMD_BT_HFP_REQUEST	65
9.3.2.2. MSG_CMD_BT_HFP_INDICATION	66
9.3.2.3. MSG_CMD_BT_HFP_CONNECTION	66
9.3.2.4. MSG_CMD_BT_HFP_RESPONSE	67
9.3.2.5. MSG_CMD_BT_HFP_STATUS_REQUEST	67
9.3.2.6. MSG_CMD_BT_HFP_STATUS_RESPONSE	68
9.3.2.7. MSG_CMD_BT_IDENTIFY_RESULT_IND	68
9.3.2.8. MSG_CMD_BT_START_IDENTIFY_REQ	68
9.4. 重要场景交互时序	69
9.4.1. 去电呼叫场景	69
9.4.2. 通话过程中发 DTMF 编码	70
9.4.3. 蓝牙状态改变场景	70
10. 通信协议	71
10.1. SERVICE TYPE 命名规则	71
10.2. 控制消息	72
10.2.1. 包结构	72
10.2.2. 相关消息体	72
10.2.3. 时序图	93
10.3. 视频消息	93
10.3.1. 包结构	94
10.3.2. 相关消息体	95
10.3.2.1. MSG_CMD_VIDEO_ENCODER_INIT	96
10.3.2.2. MSG_CMD_VIDEO_ENCODER_INIT_DONE	96
10.3.2.3. MSG_CMD_VIDEO_ENCODER_FRAME_RATE_CHANGE	97
10.3.2.4. MSG_CMD_VIDEO_ENCODER_FRAME_RATE_CHANGE_DONE	97
10.3.3. 时序图	97
10.3.3.1. 建立连接时视频消息相关时序图	97
10.3.3.2. 调整帧率时视频消息相关时序图	98
10.3.3.3. 暂停视频数据时视频消息相关时序图	99
10.4. 音频消息	99
10.4.1. 包结构	99
10.4.2. 相关消息体	100
10.4.3. 时序图	102
10.5. 语音消息	103
10.5.1. 包结构	103

10.5.2.	相关消息体.....	104
10.5.3.	时序图.....	113
10.5.3.1.	唤醒时序.....	113
10.5.3.2.	识别时序.....	114
10.5.3.3.	录音源定制时序.....	115
10.5.3.4.	VR-TTS 播报时序.....	115
10.6.	反控消息.....	116
10.6.1.	各种反控事件处理机制.....	116
10.6.1.1.	触摸屏.....	116
10.6.1.1.1.	透传方式.....	116
10.6.1.1.2.	坐标换算.....	117
10.6.1.2.	硬按键.....	117
10.6.1.2.1.	地图界面硬按键.....	121
10.6.2.	包结构.....	121
10.6.3.	相关消息体.....	122
10.6.4.	时序图.....	122
11.	手机端特殊场景策略.....	124
11.1.	手机端锁屏或者前后台切换.....	124
11.2.	手机端电话通话过程.....	125
12.	车载数据使用及建议.....	125
12.1.	SG_CMD_CAR_VELOCITY.....	129
12.2.	MSG_CMD_CAR_GPS.....	129
12.3.	MSG_CMD_CAR_GYROSCOPE.....	130
12.4.	MSG_CMD_CAR_ACCELERATION.....	130
12.5.	MSG_CMD_CAR_OIL.....	131
12.6.	MSG_CMD_CAR_DATA_GEAR.....	131
12.7.	MSG_CMD_CAR_DATA_SUBSCRIBE.....	131
12.8.	MSG_CMD_CAR_DATA_SUBSCRIBE_DONE.....	132
12.9.	MSG_CMD_CAR_DATA_SUBSCRIBE_START.....	133
12.10.	MSG_CMD_CAR_DATA_SUBSCRIBE_STOP.....	134
13.	CARLIFE 可以给车机系统提供的数据.....	135
13.1.	导航 TBT 信息.....	135
13.2.	当前歌曲播放信息.....	138
13.3.	当前歌曲播放进度信息.....	139
14.	CARLIFE 车机端退出机制.....	139
15.	更新机制.....	139
15.1.	车机端.....	139
15.2.	手机端.....	140
15.3.	版本兼容机制.....	140
15.3.1.	车机端.....	140
15.3.2.	手机端.....	140

15.3.3. 数据协议.....	140
16. 安全认证.....	141
16.1. 手机端 CARLIFE 对车机端 CARLIFE 安全认证	141
16.1.1. 认证消息定义.....	141
16.1.2. 认证流程.....	141
17. 性能指标.....	142
18. 统计信息声明	142
18.1. 车机系统统计信息声明（MSG_CMD_HU_INFO）	143
18.2. 渠道及运行统计信息声明（MSG_CMD_STATISTIC_INFO）	143
19. CARLIFE 团队对于车厂提供的支持	144
19.1. 主要测试工具	144
19.1.1. CarLife 协议工具.....	144
20. CARLIFE 认证.....	144
20.1 合作方自测并发起验收流程.....	145
20.2 百度验收	146
21. 车机端 HMI 设计规范简述.....	147
21.1. 车机端 CARLIFE ICON 及名字规范.....	148
21.2. 车机端连接动画规范	149
21.2.1. 连接前	149
21.2.2. 手机未安装 CarLife	149
21.2.3. 连接中.....	150
21.2.4. 连接成功	151
21.2.5. 连接失败/断开.....	151
21.2.6. 连接帮助	152
21.2.7. 连接中 CarLife 切到后台/手机锁屏	152
21.2.8. 退出/返回 CarLife.....	153
21.2.9. 蓝牙未连接	154
21.3. 电话功能	154
21.3.1. 蓝牙已连接	155
21.4. 车机端合作方品牌定制规范	156
21.4.1. 车机端启动页	156
21.4.2. 合作方品牌按钮	156
21.5. 默认激活	157

1. 版本历史

修改记录

版本号	修改内容简介	修改日期	修改人
V1.0	创建	2015/12/09	CarLife Team
V1.0.1	对蓝牙自动匹配和蓝牙电话章节，进行了部分修改，主要增加了初始化自动匹配的流程和一条新的命令通道消息。	2015/12/17	Yang Jie
V1.0.2	增加了蓝牙地址字段，在如下消息： MSG_CMD_BT_HFP_INDICATION， MSG_CMD_HU_INFO,MSG_CMD_MD_INFO	2016/1/5	Yang Jie
V1.0.3	增加三个命令消息， MSG_CMD_BT_HFP_RESPONSE, MSG_CMD_BT_HFP_STATUS_REQUEST 和 MSG_CMD_BT_HFP_STATUS_RESPONSE	2016/1/5	Yang Jie
V1.0.4	增加车机定制消息以及修改连接时序图 MSG_CMD_MD_FEATURE_CONFIG_REQUEST MSG_CMD_HU_FEATURE_CONFIG_RESPONSE	2016/1/8	OuYang NengJun
V1.0.5	修改 6.2.3 初始化自动匹配流程图，使用 MSG_CMD_BT_HFP_CONNECTION 来告诉车机，是否完成 Identification。同时在具体消息定义中，增加有关说明。增加电话键盘+号的按键。在 MSG_CMD_BT_HFP_INDICATION 消息的状态中，增加车机端三方通话的状态指示	2016/1/26	Yang Jie
V1.0.6	1) 修改 6.2.3 章节和流程图，添加两条新消息 MSG_CMD_BT_START_IDENTIFY_REQ,MSG_C MD_BT_IDENTIFY_RESULT_IND 到 9.3.2,以及 10.2.2 的命令列表。在 HU_INFO,MD_INFO 消息中增加蓝牙名字的字段 2) 修正 HMI 规范中启动页的描述	2016/1/29	Yang Jie Zhang Ling
V1.0.7	更新 6.2.3 蓝牙自动匹配流程图，更新 MSG_CMD_BT_START_AUTOPAIR 的描述	2016/2/1	Yang Jie
V1.0.8	增加 Bonjour 服务的名称及类型	2016/2/2	FangSheng
V1.0.9	在蓝牙初始化时序图中增加 Feature_Config 请求和响应。以及删除 MULTICALL_INACTIVE 动作，和添加 UNMUTE_MIC 命令	2016/2/25	Yang Jie
V1.1.0	修改控制消息中相关消息 service_type 值： MSG_CMD_BT_HFP_STATUS_RESPONSE\MSG_	2016/2/29	Zhao Ke

	CMD_MD_FEAUTRE_CONFIG_REQUEST\MSG_CMD_HU_FEATURE_CONFIG_RESPONSE\MSG_CMD_HU_AUTHEN_RESULT\MSG_CMD_MD_AUTHEN_RESULT		
V1.1.1	增加一个命令消息错误码统计信息 MSG_CMD_ERROR_CODE	2016/2/29	Fang Sheng
V1.1.2	修改表 10.2-2，增加用于表示音频传输方式的配置项（AUDIO_TRANSMISSION_MODE）	2016/3/4	Liu Caiquan
V1.1.3	添加视频相关消息 MSG_CMD_VIDEO_ENCODER_JPEG，MSG_CMD_VIDEO_ENCODER_JPEG_ACK	2016/3/14	Song Ye
V1.1.4	添加蓝牙电话有关消息 MSG_CMD_BT_HFP_CALL_STATUS_COVER	2016/3/16	Yang Jie
V1.1.5	完善和对齐以下消息：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	增加消息 MSG_CMD_MD_EXIT	2016/3/23	Zhao Ke
V1.1.7	修改反控透传方式传递的消息，限定为 MSG_TOUCH_ACTION 消息	2016/3/24	Zhao Ke
V1.1.8	修改 9.3.2.3 中 MSG_CMD_BT_HFP_CONNECTION 消息和实现代码不一致的地方	2016/3/30	Yang Jie
V1.2	添加 6.1.3 EAP 方案及 6.1.4 从 NCM 切换到 EAP	2016/5/24	Fang Sheng
V1.2.1	修改录音质量要求：将幅值修改到 5000 左右	2016/6/28	Ouyang NengJun
V1.2.2	增加 6.1.3.7 节 EAP 连接时车机端到手机端的心跳机制	2016/7/7	xubinluo
V1.2.3	将控制消息名称中 CarLife 改为 Carlife，Message 改位 message，并修改 NaviStatus 中状态的命名，修改控制消息章节图表编码错误	2016/07/29	Zhao Ke
V1.2.4	增加 MSG_VR_MODULE_STATUS 的描述	2016/08/30	OuYang NengJun
V1.2.5	去掉 MSG_CMD_TELE_STATE_CHANGE_INCOMING，MSG_CMD_TELE_STATE_CHANGE_OUTGOING，MSG_CMD_TELE_STATE_CHANGE_IDLE，MSG_CMD_TELE_STATE_CHANGE_INCALLING 消息，相关消息的功能由 MSG_CMD_MODULE_STATUS 来替换 删除 17 章中车机对手机的认证流程；	2016/10/31	Yang Jie Liu Caiquan

V1.2.6	根据最新的实现，更新蓝牙电话和自动匹配章节	2016/11/4	Yang Jie
V1.2.7	修改数据订阅，添加对 GPS 数据的格式支持	2016/11/10	Yang Jie
V1.2.8	修改 CarLife 验收流程全功能验收部分	2016/11/21	Long Huijuan
V1.2.9	新增 6.1.2.10 章节，介绍 CarLife 与 CarPlay 在车机端共存的拉起策略	2016/12/20	Zhang Ling
V1.3.0	新增车机端连接的统计项消息 MSG_CMD_HU_CONNECT_STATISTIC	2016/12/28	Zhou Xingjie
V1.3.1	删去 NCM 连接，IOS 建议使用 EAP 连接，Android 建议使用 AOA 连接	2017/1/23	FangSheng
V1.3.2	添加 0x0001005A~0x0001006F 预留字段；将 MSG_CMD_HU_CONNECT_STATISTIC 消息的 ID 修改为 0x00018070；添加车机类型配置项 ENGINE_TYPE	2017/3/6	Liu Caiquan
V1.3.3	修改车载 GPS 消息（MSG_CMD_CAR_GPS）去掉不需要的字段	2017/3/10	xubinluo
V1.3.4	增加根据车速禁用输入法的配置项（INPUT_DISABLE）	2017/4/11	xubinluo
V1.3.5	修改车载 GPS 消息（MSG_CMD_CAR_GPS）速度单位改成 m/s	2017/4/18	xubinluo
V1.3.6	语音通道增加语音中断消息（MSG_VR_AUDIO_INTERRUPT）	2017/4/20	Zhou Xingjie
V1.3.7	MSG_VR_MODULE_STATUS ID 冲突修改，MSG_CMD_MD_INFO 命令注释	2017/6/2	OuyangNengjun
V1.3.8	MSG_CMD_CAR_DATA_SUBSCRIBE 和 MSG_CMD_CAR_DATA_SUBSCRIBE_STOP 两个消息发送方向修正为 MD->HU。 MSG_CMD_CARLIFE_DATA_SUBSCRIBE，MSG_CMD_CARLIFE_DATA_SUBSCRIBE_DONE，MSG_CMD_CARLIFE_DATA_SUBSCRIBE_START，MSG_CMD_CARLIFE_DATA_SUBSCRIBE_STOP 四个消息的 data 部分，改为使用同样的 pb 结构——CarlifeSubscribeMobileCarLifeInfoList 以及 CarlifeSubscribeMobileCarLifeInfo	2017/6/20	Song Ye
V1.3.9	修改了 enum NaviState 的定义（两处）	2017/6/21	Liao Ruihua
V2.0.0	对 MSG_CMD_HU_PROTOCOL_VERSION 消息新增了 enum PBVersion 定义，PBVersion 最新版为 2.0 修改了 MSG_CMD_MODULE_STATUS 消息的 enum ModuleId 定义及备注 修改了 MSG_CMD_MODULE_CONTROL 消息的 enum ModuleId 定义及备注	2017/7/19	Liao Ruihua

	新增 enum EDogState 和 enum CruiseState 定义		
V2.0.1	修改 9.4.1 章节关于 Internal UI 去电场景的描述，原文档描述只适用于 iOS.	2017/7/27	Zhou Xingjie
V2.1.0	1、补充 6.1.2.4 章节，iOS App 拉起建议用带 Alert 的方式， 2、增加 6.1.2.8 章节，EAP 连接车机端数据读取方式的说明； 3、修改了 6.2 章节，修改了蓝牙自动配对的时序说明 4、补充 7.4 章关于超宽或者超高屏幕的适配策略 5、修改了 10.5 章节，关于 VR 和 VR-TTS 播放时序的说明； 6、修改第 21 章车机端验收流程说明； 7、删除 ADB 连接方式	2017/9/28	CarLife Team
V2.1.1	强调第 20 章验收标准	2018/7 / 1	CarLife Team

2. 前言

2.1. 背景

本文档是 CarLife 车机互联方案接入详细指南，为合作方在车机端接入 CarLife 的开发提供硬件、系统和软件实现等方面的详细指导。

CarLife(是一款手机车机互联方案，通过 CarLife 用户可以将移动设备（Mobile Device，简称 MD）上符合驾驶安全的应用通过多屏互动技术分享到车机（Head Unit，简称 HU）上显示，并通过车上的触摸屏、硬按键、旋钮操控和麦克风来控制 CarLife，此方案能充分利用手机和车机的各自优势，为驾驶员提供安全、丰富的信息娱乐体验。

CarLife 所有的 HMI 都经过严格的车规级优化，使得其非常适合在车上使用。
关于 CarLife 的更多信息会及时的在官网公布：<http://carlife.baidu.com>

2.2. CarLife 车机端整体架构

2.2.1. 传输介质

在 CarLife 中，移动设备和车机使用下面物理传输介质：

- USB cable（android 手机）或 iOS Lightning（iPhone 手机）：实现除电话功能之外的数据交互。CarLife 在此物理传输介质上构建了一套 tcp/ip layer。
- BT（蓝牙）：利用 HFP 协议实现电话接通、挂断等功能。
- **WiFi 直连：Android 手机支持和车机通过 wifi direct 连接，实现除电话功能之外的数据传输。**

2.2.2. CarLife 方案整体框架

CarLife 整体框架如图 2-1 所示。

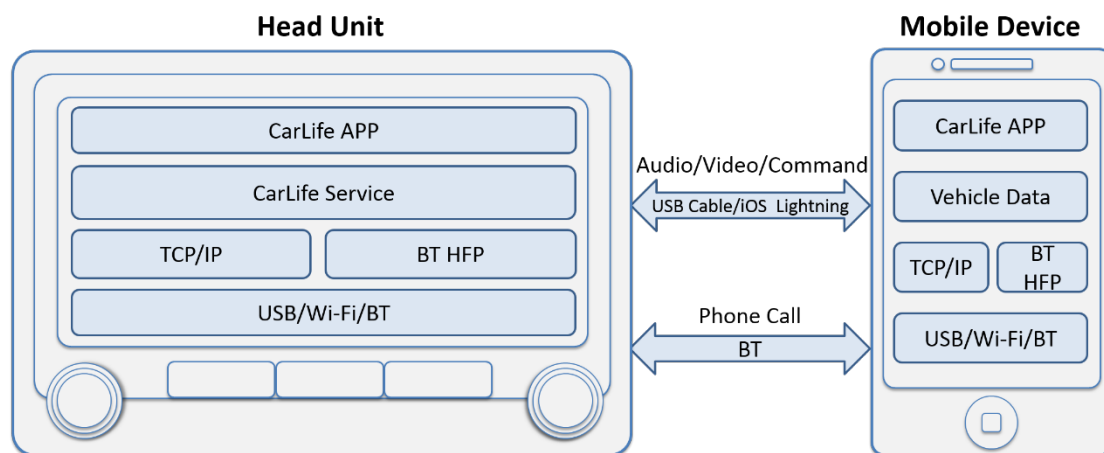


图 2.2-1：CarLife 整体框架示意图

2.2.3. 车机端架构

车机端 CarLife 由 HMI 层、CarLife 协议实现层和相应的支撑硬件组成，如图 2-2 所示：

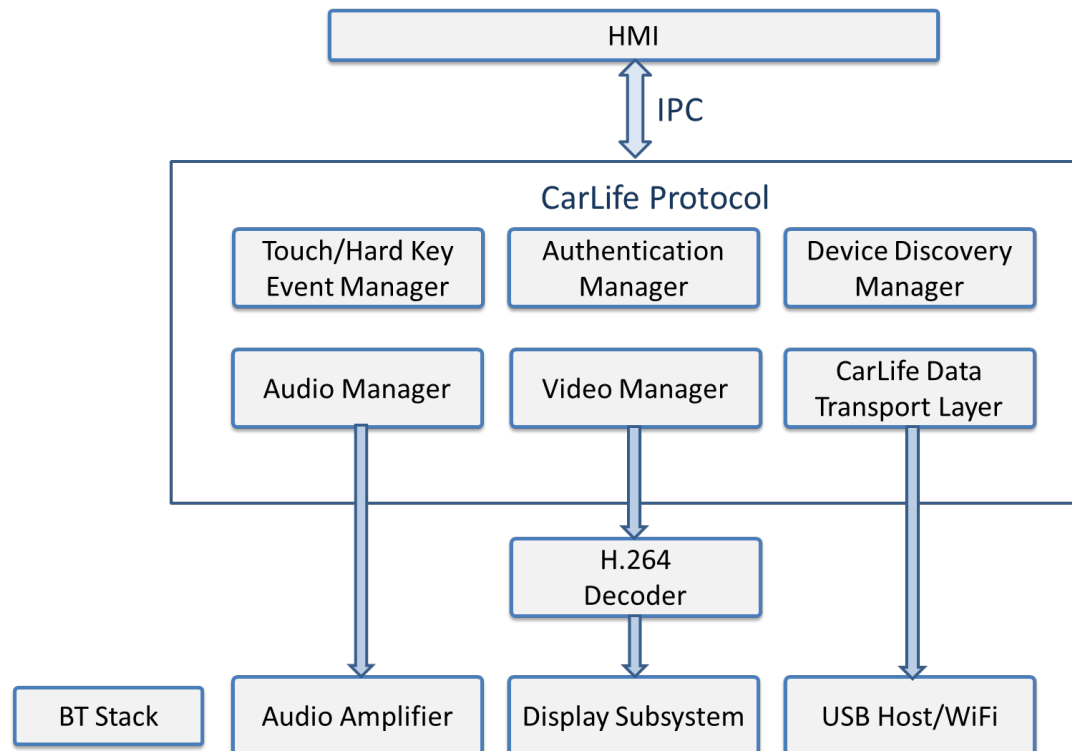


图 2.2-2: 车机端 CarLife 整体框架示意图

HMI 层负责将 CarLife 内容呈现给驾车用户，最终给使用者提供更加便捷和丰富的驾驶体验。

CarLife 协议层主要包括触摸屏/物理按键信息的获取、车机端 CarLife 与手机端 CarLife 的设备发现和鉴权管理、数据传输、视频解码和音频的播放和管理等功能。

在硬件层面，车机端 CarLife 需要获得 USB Host/Wifi、蓝牙传输、视频解码、音频播放等硬件支持。配合 CarLife 协议最终完成车机端 CarLife 的对应功能。

2.3. 通信数据的载体-PB

protocol buffer(以下简称 PB)是一种数据交换的格式，它独立于语言，独立于平台，最重要的是，PB 对于协议的兼容性有很好的支持，有力的保障了 CarLife 协议的版本兼容。

对于 CarLife 的数据协议，CarLife 官方提供了三种语言的实现：`java`、`c++` 和 `objective-C`，可方便用于各种车机端、手机端环境。由于它是一种二进制的格式，比使用 `xml` 进行数据交换快许多。可以把它用于分布式应用之间的数据通信或者异构环境下的数据交换。作为一种效率和兼容性都很优秀的二进制数据传输格式，被广泛用于诸如网络传输、配置文件、数据存储等诸多领域。

CarLife 项目中使用的 `protobuf` 版本为 2.2.0。

2.4. 名词解释

名词	解释
车机	车载娱乐信息系统（Head Unit，简称 HU）
车机端	指运行于 HU 上的 CarLife app
移动设备	Mobile Device，简称 MD，一般指 Android 手机和 iOS 手机
USB Cable	通用串行总线（英语：Universal Serial Bus，简称“USB”）是连接计算机系统与外部设备的一个串口总线标准，也是一种输入输出接口技术规范。即通常所用的 USB 线。
iOS lightning	苹果高速多功能 I/O 接口。即 iPhone 手机官方配备的数据线。
音频焦点	需要播放声音的音频模块（如音乐、导航提示音）对发声单元（speaker）的占有权。音频模块播放声音前，要先申请音频焦点，音频焦点申请成功，表示当前音频模块占用发声单元，可以播放模块声音；音频焦点申请失败，则不能进行播放；使用后可视情况释放占有权。
合作方	本文档的读者：整车厂或者供应商
量化幅值	模数转换后的数字量化数值，模拟信号经过采样变成数字信号后得到的量化数值。比如：数字量化精度是 16bit，带符号的情况下，音频模拟信号的电压最高值是 5V（参考电压 5V），模数转换后得到的数字信号数值就是 32768；音频模拟信号的电压最低值是 0V，模数转换后得到的数字信号的量化数值就是 0。

表 2.4-1 名词解释表

3. 设备要求

为了使得 CarLife 有最佳的体验，我们对于运行 CarLife 的移动设备和车机设备软硬件上都有相应的要求。

需要特别注意的是，CarLife 的最低硬件要求，覆盖了目前市面上绝大部分的设备，一般无需整车厂或供应商做任何特殊的定制。

3.1. 车机端最低硬件要求

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 mandatory (Both sides to guarantee 800x480, 60fps; For 720p 30fps)
Audio	Supported Formats:PCM Audio data Streams: - Voice stream(Mobile Device->HU)&&Navigation TTS - Media stream - Bluetooth in-call audio - VR (HU->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 mandatory for VR function: - Voice PCM flow 16bit - amplitude: about 5000

	• SNR > 12dB
--	---

表 3.1-1 车机端最低硬件要求

3.2. 手机端要求

- Android 手机：Android4.1 及以上
- iOS 手机：支持 iPhone5 及以上，iOS 系统要求 iOS8.0 及以上

4. 车机端与 CarLife 关系密切的能力

- H.264 硬件解码器
当 CarLife 在运行过程中，车机端会使用到车机上的 H.264 硬件解码接口用于视频解码。
- 麦克风
当进行语音功能的时候，CarLife 会使用车机上的麦克风进行录音。如果车机端不支持这个功能，需要通过相关的协议消息体通知手机端，对应协议请参见《12.5 语音消息》。
- Speakers
当进行音频播放、电话功能时，CarLife 会使用到车机上的 Speaker。
- 原生的电话界面
如果车机端选择集成 CarLife 的蓝牙电话功能，那么在使用 CarLife 电话功能时，要求车机把原生的电话相关的 HMI Block 掉，统一由 CarLife 来接管车机电话 HMI。当用户不启动 CarLife 时，原生电话 HMI 不受此影响。
- 音频焦点
在 CarLife 音频引擎中，会根据不同的场景进行车机音频焦点的申请和释放。

5. 数据通道

CarLife 协议规定了五种数据流，控制流（Cmd）、视频流（Video）、音频流（Audio）、语音流（VR）以及反控流（Touch），其中将音频流分为 Media 和导航 TTS 两种类型。协议定义了以下六个 TCP socket 通道用于传输这些数据流：

- 控制流（Cmd）：1 个 tcp socket 通道，主要在车机和移动设备之间传输相关控制消息，比如车载数据、蓝牙自动配对消息、视频初始化消息等。
- 视频流（Video）：1 个 tcp socket 通道，主要用于移动设备向车机传输视频编码数据。
- 音频流（Audio）：2 个 tcp socket 通道，一个用于传输 Media 音频流，另一个用于传输导航 TTS（Text to Sound）音频流，分别用于移动设备向车机传输音乐和导航语音播报的 PCM 流。

- 语音流（VR）：1 个 tcp socket 通道，主要用于车机向移动设备传输车机 mic 录制的用户语音数据。
- 反控流（Touch）：1 个 tcp socket 通道，主要用于传输车机触摸事件或者按键事件到移动设备端。

不同的连接方式，建立上述六个通道的方法不太一样，在连接引擎章节中将详细描述。

6. 连接引擎

6.1. 连接方式

6.1.1. AOA 连接

6.1.1.1.AOA 简介

针对 Android5.0 系统以及以上版本的移动设备，使用 AOA（Android Open Accessory）进行连接。AOA 是 Google 为 Android 系统提供了一种外设通讯机制。该机制可以通过蓝牙或者 USB 接口和外设通讯，无需 Android 系统特殊系统权限，包括 root 和 adb debug，对于 CarLife 而言，移动设备端不需要获取 root 权限和打开 adb debug 调试模式。但是，外设需要有操作 USB 端口的权限，对于 CarLife 而言，车机端需要有操作 USB 端口的权限。AOA 在 Android 3.1（API level 12）及更高的平台中直接支持，作为一个附加库也被 Android 2.3.4（API level 10）支持。AOA2.0 在 Android4.1（API level 16）及更高的平台中直接支持。

在 Android 移动设备上，Android 系统提供标准的 Accessory API 用于访问外设。采用 USB 通讯时，外部设备的 USB 角色为 Host，Android 移动设备为 Device。

Google 官方提供了 Android Open Accessory Protocol 的定义：

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

Google 官方提供了 adk2012 开发套件（包含 Android 端和 Accessory 端的源码，Accessory 的参考设计为 Arduino 平台）：

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

当前 5.0 以上的手机占比居高，ADB 与 AOA 连接建议默认使用 AOA 连接，或者优先使用 AOA 连接，连接成功后首先检测手机系统的版本号，如果手机系统版本低于 5.0 则主动切换到 ADB 模式。

6.1.1.1.1. Android USB 通信模式

Android 系统支持多种 USB 外围设备。根据 Android 设备在 USB 通信中充当的角色，可以将 Android USB 通信分为主机模式（Host Mode）和配件模式（Accessory Mode）两种模式。详见 Google 官方链接：

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

➤ 主机模式

主机模式是指 Android 设备充当 USB 主机并为总线供电。此模式下，Android 设备需支持 USB 主机功能或 OTG 功能，此时 Android 设备的 USB 主机称为 USB 嵌入式主机 EH（Embedded Host）。与 PC 上的 USB 主机相比，EH 设备可能无法为连接到其总线上的未识别外围设备加载驱动程序，因此它们对其目标外围设备列表 TPL（Target Peripheral List）进行了定义。这些外围 USB 设备大部分为 HID 设备（Human Interface Device）、BOMS 设备（Bulk Only Mass Storage，如 U 盘）和 CDC 设备（Comm-unication Device Class，USB 通信设备类，如打印机），其驱动程序已存在于 Android 平台的系统中（Linux Kernel），因此 Android 设备可以与其直接通信。

➤ 配件模式

配件模式是指 Android 设备充当 USB 从机，外部设备充当主机并为总线供电。此模式下，外部 USB 设备称为 Android 配件。该模式为不具备主机功能的 Android 设备提供与 USB 设备交互的能力。Android 设备和 Android 配件都必须支持 AOA 协议。不支持 AOA 协议的设备可以通过 Android 配件开发板（ADK 板）与 Android 设备连接，成为 Android 设备的间接配件。

主机模式和配件模式示意图如图 6.1-1 所示。

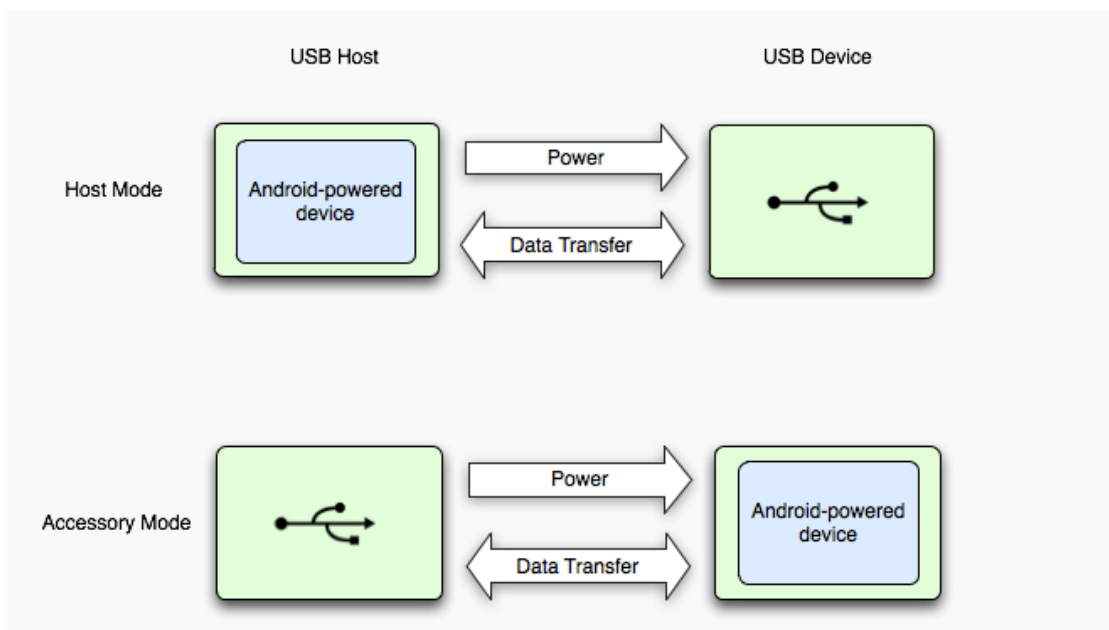


图 6.1-1：主机模式和配件模式示意图

主机模式和配件模式在 Android 3.1(API level 12)及更高的平台中直接支持。配件模式作为一个附加库也被 Android 2.3.4 (API level 10) 支持。设备厂商可以选择是否在设备的 Linux 系统镜像中包含附加库。对主机和配件模式的支持最终取决于外部设备的硬件，而不是 Android 设备的平台等级。

6.1.1.1.2. Android 配件、ADK 以及 AOA 协议

Android 配件是指遵循 AOA 协议的硬件，是 Android 配件模式中的概念。从硬件角度讲，Android 配件具有实现 USB EH 功能和 AOA 协议的微处理器以及包括 USB 接口在内的输入/输出接口。Android 配件必须与 Android 平台一起工作。

ADK (Accessory Development Kit) 是开发 Android 配件的参考开发包，该开发包基于 Arduino 公司的开源电子成型平台。ADK 开发包包括 Android 配件硬件设计文档、Android 配件固件源码、Android 设备程序源码。

AOA 协议是 Android 设备与 Android 配件之间进行 USB 通信需遵循的连接协议，该协议规定了 Android 配件如何检测 Android 设备并与其建立通信。AOA 协议规定配件需具备以下 4 项功能：

- 1、侦听自身的 USB 接口，等待与其连接的 Android 设备；
- 2、检测 Android 设备是否支持 AOA 协议；
- 3、如果需要，启动 Android 设备的 USB 配件模式；
- 4、与 Android 设备建立 USB 通信。

6.1.1.2.AOA 设备握手

6.1.1.2.1. AOA 设备握手过程

AOA 协议规定 Android 设备和配件之间握手的大致过程如下：

- (1) 通过 USB 线连接 Android 设备与配件。

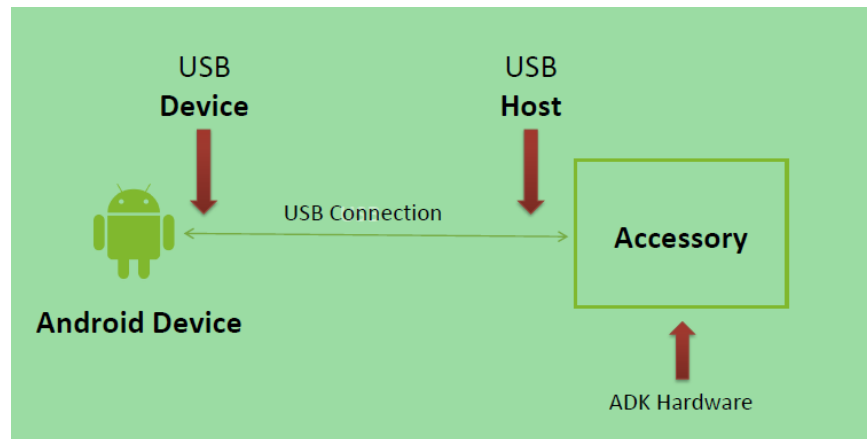


图 6.1-2: AOA 设备握手过程示意图 1

(2) 配件枚举连接过来的设备。根据 VID 和 PID 判断当前 Android 设备是否处于 Accessory Mode。如果否，则配件会向 Android 设备发出切换到 AOA 模式的请求，进入第 (3) 步；如果是，则直接调转到第 (5) 步。

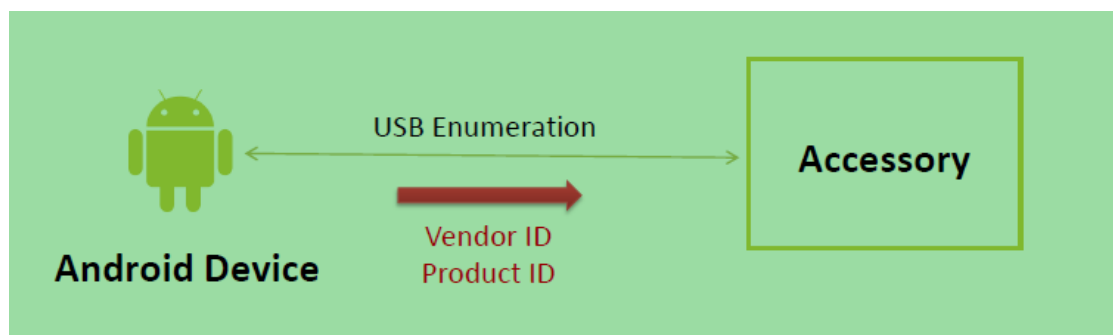


图 6.1-3: AOA 设备握手过程示意图 2

判断 Android 设备处于 Accessory Mode 时，VID 和 PID 值必须满足如下条件如下：

类别	值	备注
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

表 6.1-1 Android 设备处于 Accessory Mode 的 VID 和 PID 值

(3) 配件发送查询指令给 Android 设备。绝大多数 Android 设备，在缺省情况下都不挂载 Accessory 驱动，即不会默认处于 Accessory Mode，在配件与 Android 设备建立 USB 连接时，配件会通过握手协议查询该设备是否为 Android 设备且是否支持 AOA 协议以及支持的版本号。配件向 endpoint 0 端口中写入 51 号指令，如果返回值为 1 或者 2，则说明 Android 设备支持 AOA1.0 或者 AOA2.0，如果小于 1 或者大于 2，则说明连接设备不支持 AOA 协议或者支持的 AOA 协议版本号不正确。

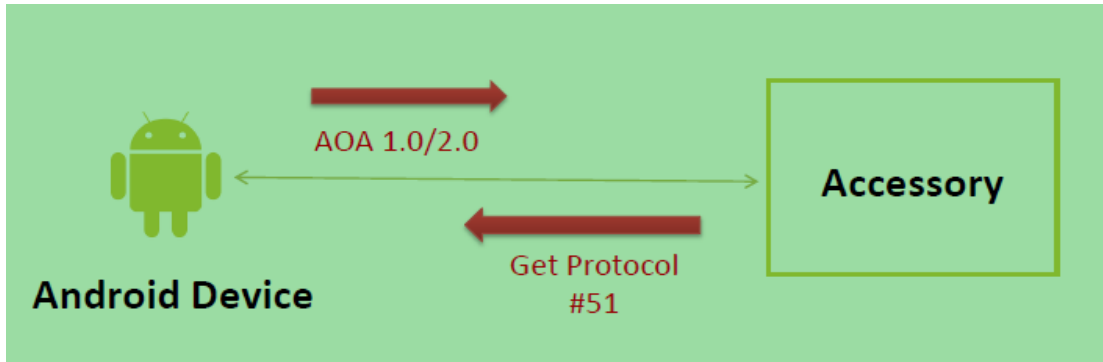


图 6.1-4：AOA 设备握手过程示意图 3

(4) Android 配件发送认证信息给 Android 设备，并发送开始通信的指令。

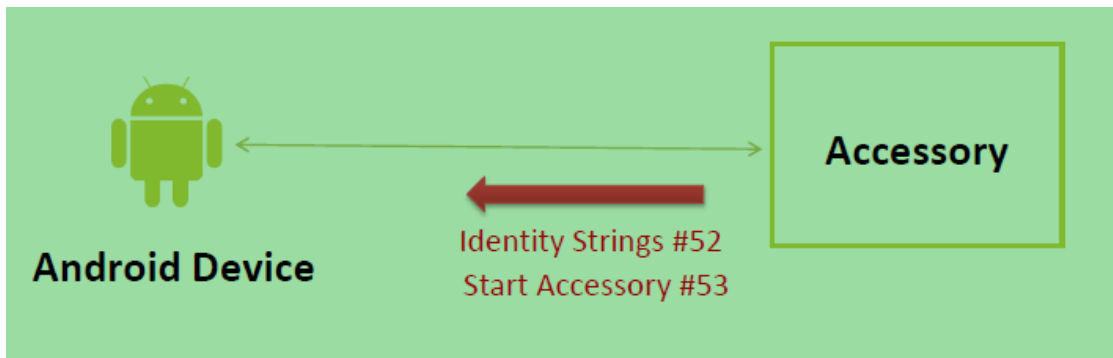


图 6.1-5：AOA 设备握手过程示意图 4

配件向 Android 设备写入 52 号指令，并附带如下一些信息。

编号	内容	实例
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.

表 6.1-2：AOA 握手附加信息

配件也会向 Android 设备发出 53 号指令，请求 Android 设备切换到 AOA 模式，Android 设备会执行请求，将 USB 切换到 AOA 模式；在这个过程中，USB 连接会出现一次逻辑插拔，配件会重新枚举设备，进入第（2）步。Android 设备接收到配件通过 52 指令发送过来的参数信息以后，使用其中的三个参数 Manufacturer、Model Name、Version 来确定绑定到该配件的 App。如果系统内无任何 App 可以匹配配件设备发来的上述三个参数，则 Android 设备会弹出一个对话框，向用户提供 Accessory 设备发送过来的描述信息和 URL 信息，用户可以点击 URL 访问它指向的 Web 页面，如下图。

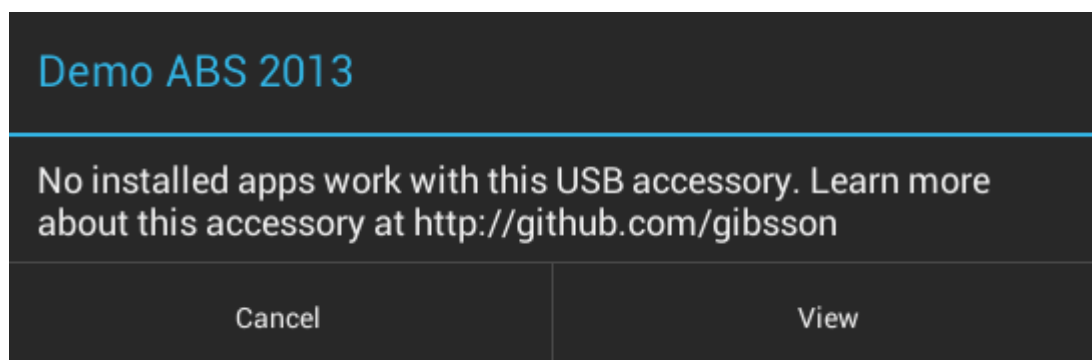


图 6.1-6：AOA 设备握手过程示意图 5

如果系统内有 App 可以匹配 Accessory 设备发来的握手信息，则 Android 系统会弹出一个对话框询问用户是否立刻启动该 App。如果用户选择 OK 则启动该 App；同时该对话框提供一个勾选框，勾选之后每次 Accessory 设备连接后会自动启动该 App。应该要求用户勾选该对话框，否则 App 启动后向 USB Manager 获取 Accessory 设备后可能因为 Permission 问题无法打开文件描述符建立通讯连接。如下图。

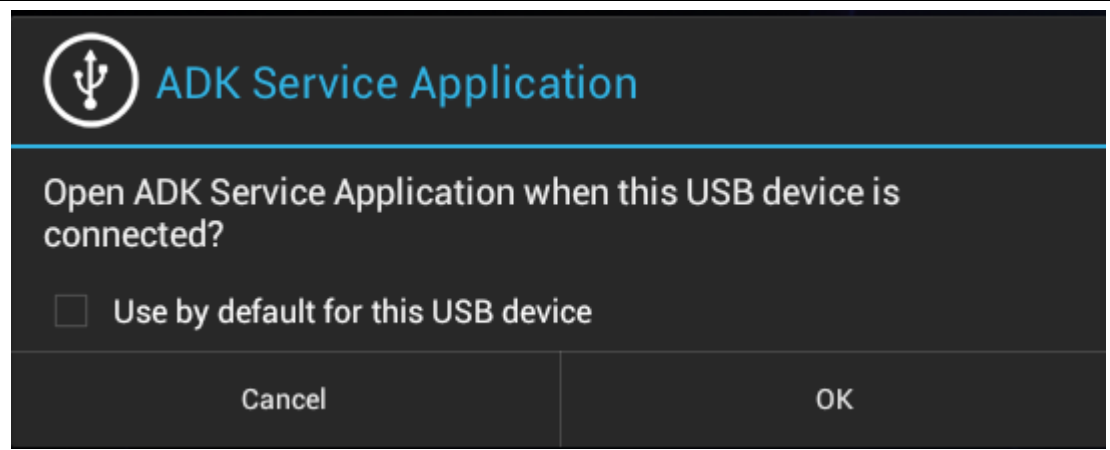


图 6.1-7：AOA 设备握手过程示意图 6

(5) 建立连接

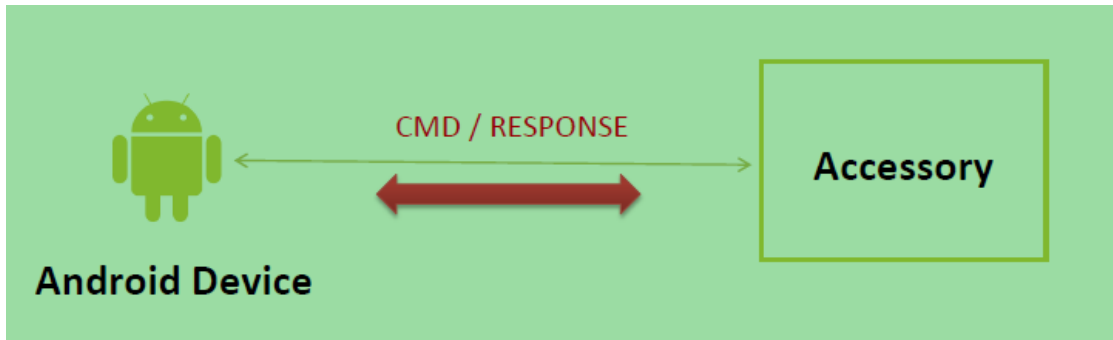


图 6.1-8：AOA 设备握手过程示意图 7

6.1.1.2.2. AOA 设备握手指令

指令编号	作用	CarLife 目前是否使用
51	判断设备是否支持 AOA 协议，并获取支持 AOA 协议的版本号。	是
52	发送配件相关设备认证信息，Android 设备可以据此来确定绑定到该配件的 APP。	是
53	请求 Android 设备切换到 Accessory Mode	是
54~57	HID 支持	否
58	Audio 支持	否

表 6.1-3：AOA 设备握手指令

6.1.1.2.3. AOA 设备握手流程图

综上所述，AOA 的设备握手流程如下图。

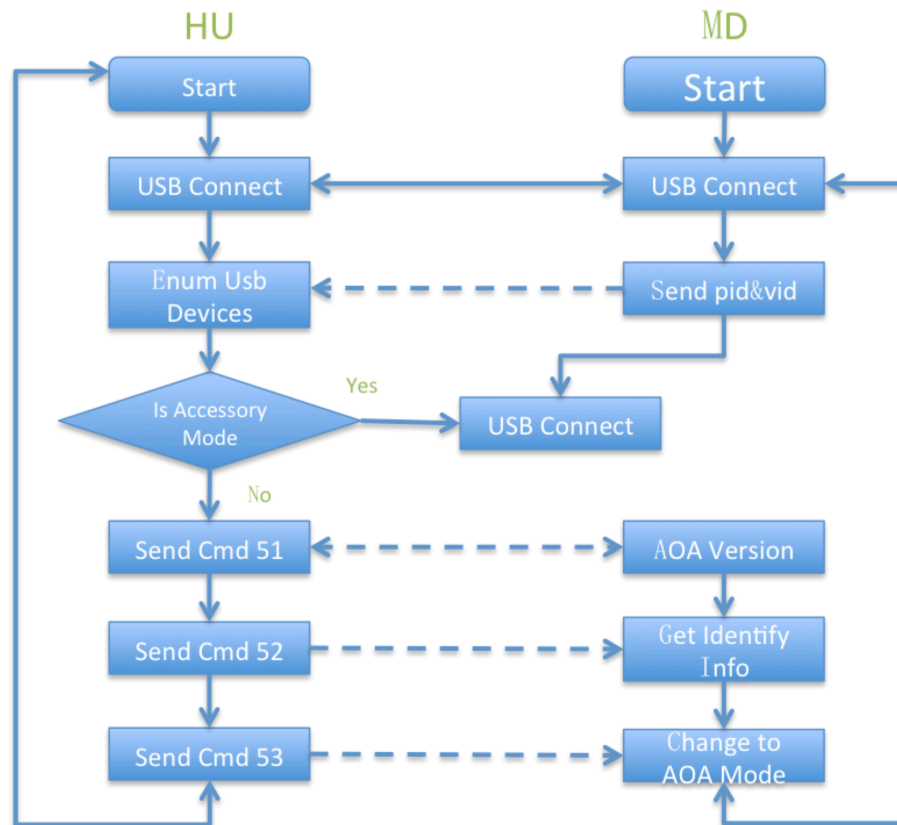


图 6.1-9： AOA 的设备握手流程图

6.1.1.3.AOA 消息结构

为了不对其他模块（比如音视频、语音、地图等模块）产生影响，尽量减少代码开发的工作量以及对其他连接方式的兼容，AOA 连接各个通道的消息结构保持不变，详见第 10 章节通信协议。由于 AOA 实际上在传输的过程中只有一个通道，因此需要在原消息结构的基础上增加一个头用来标识该条消息是属于哪一个通道。具体结构如下图所示。



图 6.1-10： AOA 连接包结构示意图

如上图，AOA Head 总占 8 个字节，Channel ID 和 Msg Len 各占 4 个字节，具体如表 8-3。

表 6.1-4： AOA 连接包结构各字段含义

字段名称	位偏移	大小(byte)	说明
Channel ID	0~3	4	标识该消息是属于哪个通道，Cmd, Video, Media, TTS, VR 以及 Touch 通道一次为 1， 2， 3， 4， 5， 6
Msg Len	4~7	4	后面 CarLife Msg 的整体长度
CarLife Msg	8~n	n-8	CarLife Msg 消息结构， 详见 通信协议 章节

车机端在发送 AOA 数据时需要注意将 8 个字节的 AOA Head 和 CarLife Msg 单独分开发送。

经过 6.1.1.2 小节介绍，在车机端执行 51、52、53 号指令以后，车机端重新扫描当前连接到车机的 USB 设备，此时移动设备端上报的 USB 端口为 AOA 端口，由此得到 UsbDevice 和 UsbDeviceConnection 两个对象，据此进行输入输出操作；移动设备端对应的 CarLife 程序会被自动拉起并获取得到 UsbAccessory 对象，并由此得到 FileOutputStream 和 FileInputStream，据此进行输入输出操作。详细的设计请参考 Android 官方文档。

6.1.1.4.AOA 车机端开发

由于目前有 Android、Linux、QNX 以及 WinCE 等不同系统的车机，在 Android 系统中，Android 官方是提供 USB 的相关接口用来进行 AOA 连接，主要有 UsbManager, UsbDevice, UsbInterface, UsbDeviceConnection 等。对于其他非 Android 车机没有以上接口，只能使用 libusb 等开源库直接对 USB 设备进行操作，包括发送相关控制命令指令以及传输数据等，具体的流程和在 Android 系统下使用 Android 提供的接口类似。

6.1.1.5.AOA 数据通道建立

当使用 AOA 将移动设备和车机建立连接以后，建议使用如下方式来和移动设备端建立数据通道：在车机端将各模块和底层数据收发模块之间建立六个 socket 用于通信。。当使用 AOA 或者 libusb 等输入接口从 usb 端口中读入数据以后，进行拆包（去除 AOA 头，8 个字节），得到 CarLife Msg 结构（详见第 13 章节），根据 Channel ID 分发到对应通道的 socket 中，其他模块（比如 Cmd、Video、Media 等）可以从 socket 中读入数据并对 CarLife Msg 进行解析，获取具体的数据。相反，当其他模块需要向移动设备端传输数据时，往对应的 socket 中写入数据，AOA 连接模块通过 socket 接收数据以后，加上 AOA 头（Channel ID 和 Msg Len 共计 8 个字节）调用输出接口写入到 usb 端口中发送给移动设备端。这种方式利用 socket 实现了消息缓存机制，并且将底层连接模块和 CarLife 上层模块之间充分解耦。如图 8-11 所示：

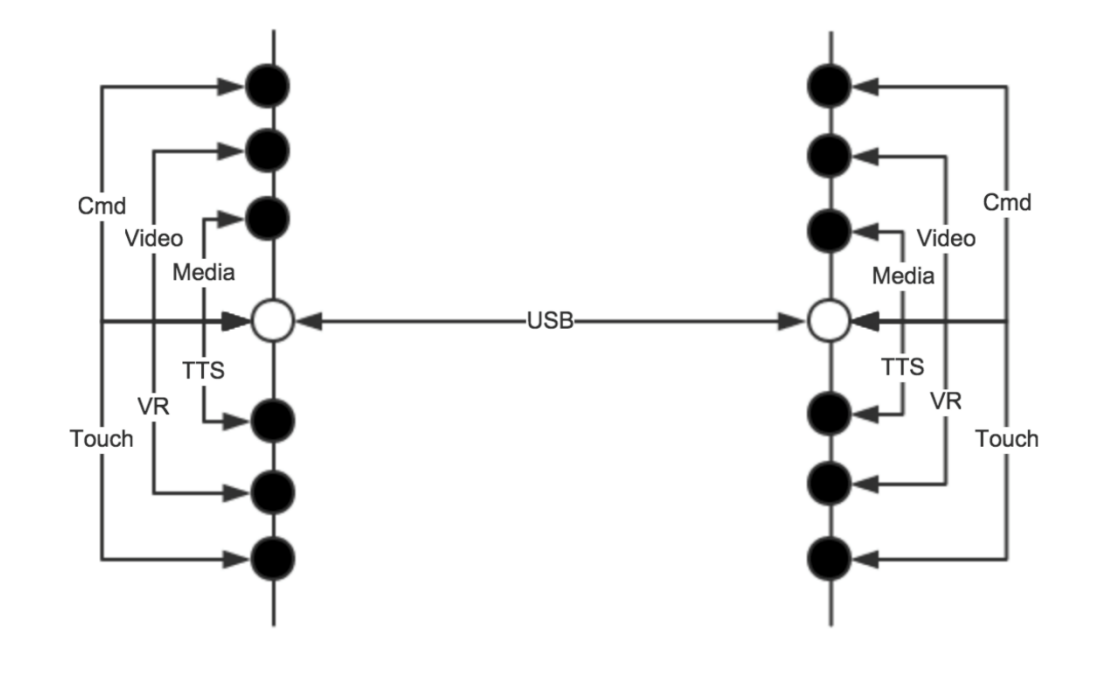


图 6.1-11： AOA 连接数据通道建立示意图

6.1.1.6.车机端 CarLife AOA 连接动画百分比定义

车机端 CarLife 在连接中需要有连接动画（请参考 HMI 规范章节中的连接中），对于连接进度的百分比建议采用以下方式分配，便于用户反馈问题的定位：

百分比值（%）	备注
10	检测到 Android 手机设备后
30	USB 初始化（如获取 interface 等）
40	检测是否处于 AOA 模式

50	切换到 AOA 模式
60	获取 AOA 的两个 EP (EndPoint)
80	完成单通道模拟 6 个 socket
100	6 个 socket 连接成功

表 6-2: AO 连接中的百分比

6.1.1.7. 移动设备端 CarLife 应用拉起与判别

6.1.1.7.1. 移动设备端 CarLife 拉起

AOA 连接方式下，各种状态下拉起手机端 CarLife 的方式如下表：

连接状态	手机 CarLife 运行状态	拉起方式
连接过程中	运行（在后台）	AOA 的 52 号指令
连接过程中	未运行	AOA 的 52 号指令
已连接	运行（在后台）	使用 MSG_CMD_GO_TO_FOREGROUND 控制消息

6.1.1.7.2. 移动设备端是否安装 CarLife 应用

如果当前移动设备中没有安装 CarLife，AOA 连接时会弹出对话框，提示用户。目前车机端没有太好的方式判断移动设备端是否安装 CarLife。一种可能的方式是当车机端检测到使用 AOA 连接时一段时间内收不到数据时，在车机端显示移动设备端 CarLife 应用程序下载地址（<http://webpage.navi.baidu.com/static/webpage/carlife/carlifedownload.html>）的二维码，此地址不会改变，但页面上的下载链接会由百度根据手机端版本发布情况动态更新。移动设备（Android & iPhone）访问这个页面会直接下载最新的 CarLife 手机端。

6.1.2. EAP 连接

EAP 即 External Accessory Protocol，是苹果官方提供的外设和 iPhone 手机通信协议。EAP 主要包括两种方式：EA Session 和 EA Native Transport，EA Session 主要是用来传输 iAP2 的控制消息，传输速率比较低；EA Native Transport 可以上报包含两个 EP 的 interface，其传输速率比较高效，CarLife 的 EAP 方案就是采用了这种方式。该方案参考了苹果官方文档《MFi Accessory Interface Specification R23》，下文在引用时以 R23 代替。

6.1.2.1.车机端的开发工作

车机端关于 iPhone 有线连接的开发工作主要包括：**MFI 认证**、**USB 驱动**、**数据读写**、**数据通道的建立**。其中 **MFI 认证** 及 **USB** 都和硬件密切相关，单纯的应用是无法完成的，所以需要车机必须具备一定的软硬件条件，建议流程如下：

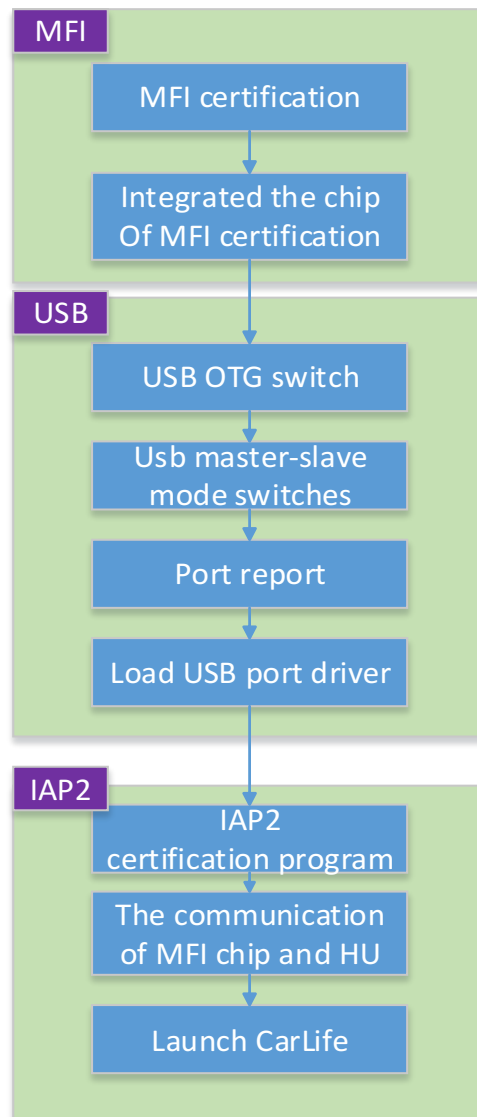


图 1-1：车机端开发工作流程

1、MFI 认证

车机厂商需要加入 MFI，并在苹果 MFi 官方网站<mfi.apple.com>上面提交产品计划，申请 MFi 认证。

2、MFI 认证芯片的集成

通过 MFI 认证后，购买苹果提供的加密芯片，并集成在车机上。

3、OTG 切换

车机 USB 端口需要支持 OTG 切换，连接 Android 手机的时候车机是主模式，连接 IOS 手机的时候车机会从主模式切换到从模式。

4、主从模式切换

iPhone 手机插入到车机后，车机先做为 HOST 对 iPhone 手机进行枚举，检测到手机是 ios 手机后车机会对手机发送 role switch 的 USB 请求，等待手机对车机进行枚举，枚举完后车机做为从设备。

5、端口上报

车机端主从模式切换后，需要给手机上报 iAP2 及 EA Native Transport 端口。iAP2 主要是做数据的鉴权，EA Native Transport 端口主要是用来传输 CarLife 的数据信息。

6、USB 端口驱动

车机端给手机上报完 iAP2 及 EA Native Transport 端口后，需要加载对应的驱动

7、IAP2 认证

iPhone 手机与车机在进行数据交互前需要通过苹果的认证，即根据苹果的认证协议(R23)通过 MFI 认证芯片与手机进行 IAP2 的鉴权认证。认证通过后 CarLife 的数据才可以通过 USB 端口进行传输。

8、认证芯片与车机主芯片交互

MFI 认证芯片是通过 I²C 总线与车机主芯片进行交互的，加载完 MFI 认证芯片的驱动后，在系统设备文件中能够映射 MFI 芯片的设备文件，IAP2 认证程序通过该文件与 MFI 芯片通信。

9、拉起 CarLife

拉起 CarLife 是通过苹果的协议 app launch 来完成的。请参考 R23 中 App Launch

6.1.2.2.USB 连接层开发

6.1.2.2.1. 连接过程

1、连接的整体流程

iPhone 通过 USB 连接车机的流程图如下所示：

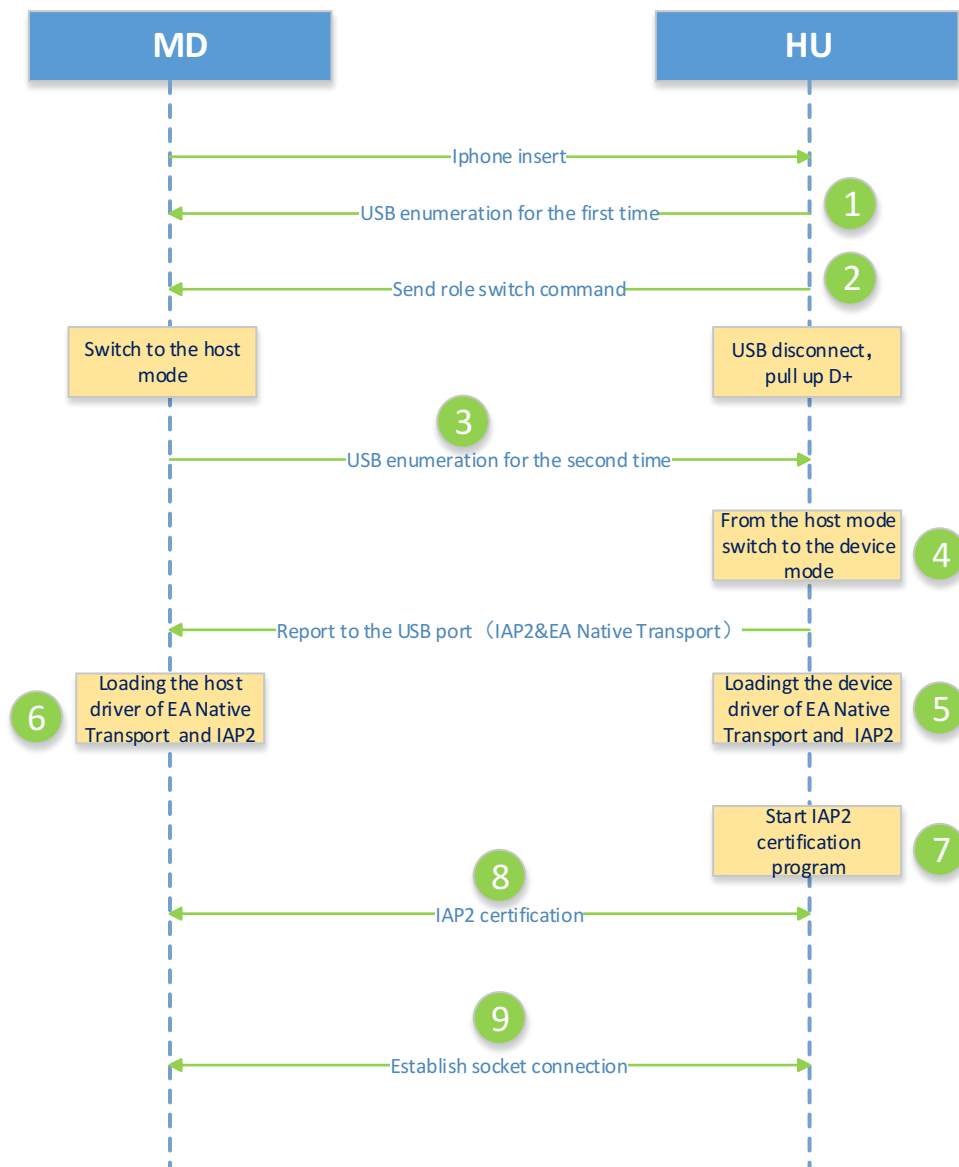


图 2-1：连接流程图

(1) 车机做为 HOST 枚举手机

iPhone 手机刚插入车机时，车机做为 HOST 会对手机进行枚举。车机通过枚举可以获取手机的端口信息，并根据这些信息判断当前是否是 iPhone 手机。

(2) 发送 role switch 命令

车机通过手机的 ep0 向手机发送 role switch 切换命令，并且主动断开 USB 连接，手机收到命令后，USB 会 reset 并切换到 HOST 模式。

(3) 手机做为 HOST 枚举车机

手机切换到 HOST 模式后，对通过 USB 连接的车机进行重新枚举。

(4) 车机模式切换

经过手机的枚举后，车机会切换成 device 模式并向手机上报标准的 EA Native Transport 端口及符合苹果认证协议的 IAP2 端口

(5) 车机端加载设备驱动

车机端加载 EA Native Transport（在车机端映射的设备文件/dev/mfi-ean）、IAP2 从设备

的驱动，还会加载 MFi 芯片(主要是通过 I²C 同车机交互)的驱动，完成后在 dev 目录下映射相应的设备文件（如/dev/MFi）。

（6）手机端加载 HOST 驱动

手机端会加载 USB 的 HOST 驱动，这部分苹果已经完成。

（7）车机启动 IAP2 认证程序

（8）IAP2 认证

根据苹果的 iAP2 认证协议开发其认证程序，该程序通过/dev/MFi_iAP2（IAP2 驱动映的设备文件）与手机交互，通过/dev/MFi（加载 MFi 认证芯片驱动映射的设备文件）与认证芯片进行数据交互。

（9）建立 6 个 socket 通道

需要将 EA Native Transport 端口虚拟成 6 个 socket 通道，建立 6 条 TCP Socket 通道，分别是：控制流（CMD）双向数据传输；视频流（Video）单向数据传输，由移动设备传输到车机；音频流（Audio）2 个 TCP socket，一个用于传输 Media 音频流，另一个用于传输 TTS 音频流，均为单向数据传输，由移动设备发送到车机；语音流（VR）双向数据传输，由车机传输到移动设备；反控流（Touch）单项数据传输，由车机传输到移动设备。

2、EA Native Trasport

使用 EA Native Transport 的方式进行传输，手机必须是主模式。并且车机需要上报的两个 interface，两个 interface 的选择是手机端的 APP 决定的。当 APP 打开 EA session 的时候需要切换到具有两个 EP（bulk IN and OUT）的端口进行通信，当其关闭的时候能够切到没有 EP 的 interface。下面是一个 EA Native Transport 的例子仅供参考：

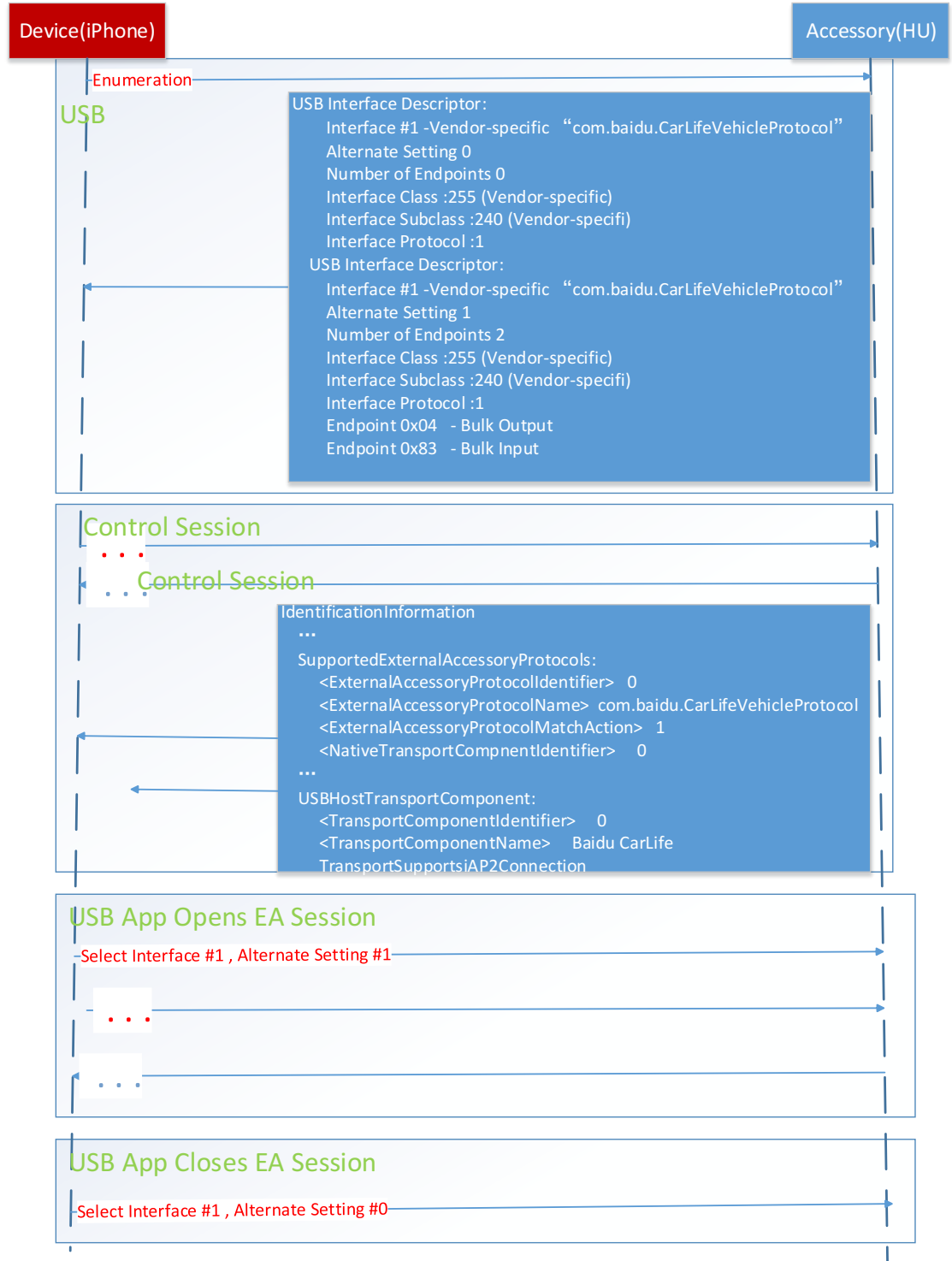


图 2-2:EA Native Transport 流程图

6.1.2.2.2. 模式切换

1、模式切换过程

iPhone 手机插入车机后，车机会对其进行枚举。当检测到苹果手机便会发送模式切

换命令，如下流程图：

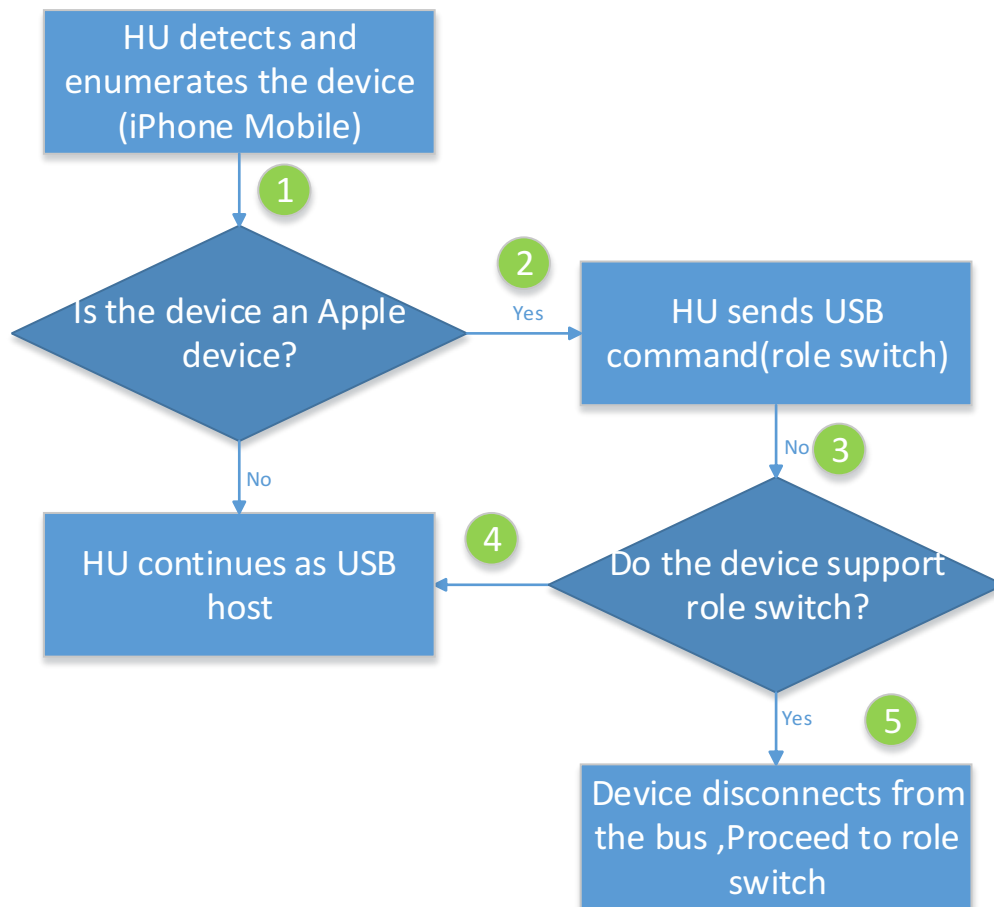


图 2-3：USB 模式切换流程

- (1) 手机插入车机后，车机会根据枚举的信息（设备 VID、PID）来判断当前是什么类型的手机，如果不是苹果手机仍然保持 HOST 模式
- (2) 检测到是苹果手机，车机会通过 USB 的 EP0 给手机发送 role switch 命令，来切换模式
- (3) 手机收到 role switch 命令后，首先会判断是否支持
- (4) 手机不支持 role switch 命令，HU 不会做任何处理
- (5) 手机支持 role switch 命令，车机会主动断开与手机的 USB 连接，并拉到 D+ 等待手机来重新枚举。

2、模式切换命令

车机检测到 iPhone 手机插入后，会通过 EP0 给苹果手机发送如下命令来切换 iPhone 的主从模式，其命令格式为：

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

表 2-1: 模式切换命令

6.1.2.2.3. 上报的 USB 端口

车机在切换到从模式时，需要向手机上报两个 USB 端口：EA Native Transport、IAP2 端口。分别用来做 CarLife 的数据传输和苹果的 IAP2 认证。

1、EA Native Transport 端口

EA Native Transport，它有两个 Interface：一个 Interface 具有两个 EP，分别是 bulk IN 和 bulk OUT，当进行数据传输的时候必须选择这个 interface；一个 Interface 是没有 EP 的，是不能进行数据传输的。端口属性如下：

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 ExternalAccessoryProtocolName
Number of Endpoints	2	one bulk In and one bulk OUT endpoint
Alternate Setting	1	

表 2-2: 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.CarLifeVehicleProtocol	Must match an identified ExternalAccessoryProtocolName
Number of Endpoints	0	one bulk In and one bulk OUT endpoint
Alternate Setting	0	

表 2-3: EA Native Transport Alternate Setting 0

2、IAP2 端口

车机要想和处于主模式的 iPhone 手机建立 iAP2 的通信，就必须上报一个特殊的 USB 端口（两个 EP：bulk out & bulk in）即 iAP2 端口。

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

表 2-4: IAP2 端口信息

处于主模式的 iPhone 手机通过 bulk out EP 给车机发送数据，车机如果正确收到数据需要给 iPhone HOST 端回复一个 USB ACK 包，数据错误或者车机处于非 ready 状态需要回复一个 USB NAK 包。

6.1.2.3. 鉴权

车机和 iPhone 手机之间建立通信需要经过认证芯片进行鉴权，当前认证芯片的版本主要有 2.0B 和 2.0C，官方推荐集成 2.0C 版本。手机、车机和认证芯片的交互流程如下所示：

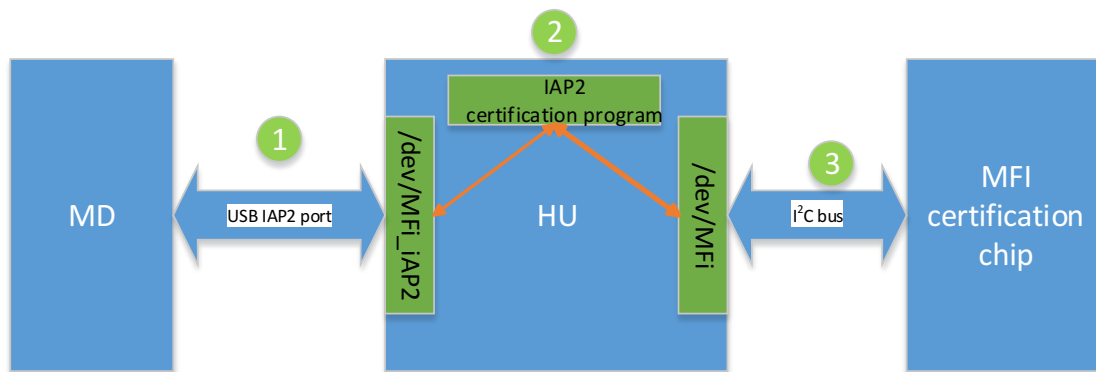


图 3-1: 鉴权流程

手机和车机之间的鉴权指令都是通过 IAP2 这个端口进行传输的，这个端口是车机切换成从模式时上报给手机的。车机在 USB 端口驱动加载完成后会启动 IAP2 认证程序，该程序通过 IAP2 端口映射的设备文件（/dev/MFi_IAP2）与手机通信，通过 MFI 认证芯片加载驱动过程中映射的设备文件与 MFI 认证芯片进行通信。MFI 认证芯片和车机之间是通过 I²C 总线连接，其传输遵循 I²C 协议。

6.1.2.3.1. Accessory Authentication

IAP2 的认证程序是根据苹果的官方文档 R23(12 章 Accessory Authentication)来完成的，主要涉及到的鉴权认证指令如下所示：

1、RequestAuthenticationCertificate

其命令参数如下：

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

表 3-1: RequestAuthenticationCertificate 命令

Accessory(HU)需要在 1 秒内对 RequestAuthenticationCertificate 命令做出相应，具体的响

应见命令 2。

2、AuthenticationCertificate

其命令参数如下：

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

表 3-2: AuthenticationCertificate 命令

如果证书有效，Apple device 将会继续发起 RequestAuthenticationChallengeResponse 命令，Accessory 需要通过授权芯片计算出 Authentication Challenge 的 response，并在 2 秒内发送消息给 Apple device，具体的命令参考 3 和 4

3、RequestAuthenticationChallengeResponse

其命令参数如下：

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

表 3-3: RequestAuthenticationChallengeResponse 命令

4、AuthenticationResponse

其命令参数如下：

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

表 3-4: AuthenticationResponse 命令

5、AuthenticationFailed

其命令参数如下：

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

表 3-5: AuthenticationFailed 命令

6、AuthenticationSucceeded

其命令参数如下：

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

表 3-6: AuthenticationSucceeded 命令

除了以上命令外还用到 StartIdentification、IdentificationInformation、IdentificationAccepted、IdentificationRejected、CancelIdentification、IdentificationInformationUpdate 命令。具体使用请参考苹果认证协议文档中的 Accessory Identification 章节。

车机（HU）收到 iPhone 设备发送过来的认证指令后会给 MFI 认证芯片发送请求，MFI 认证芯片根据具体的指令算法处理并将结果返回给车机的 IAP2 认证程序，然后再返回给手机。下面是一次成功认证过程时序图：

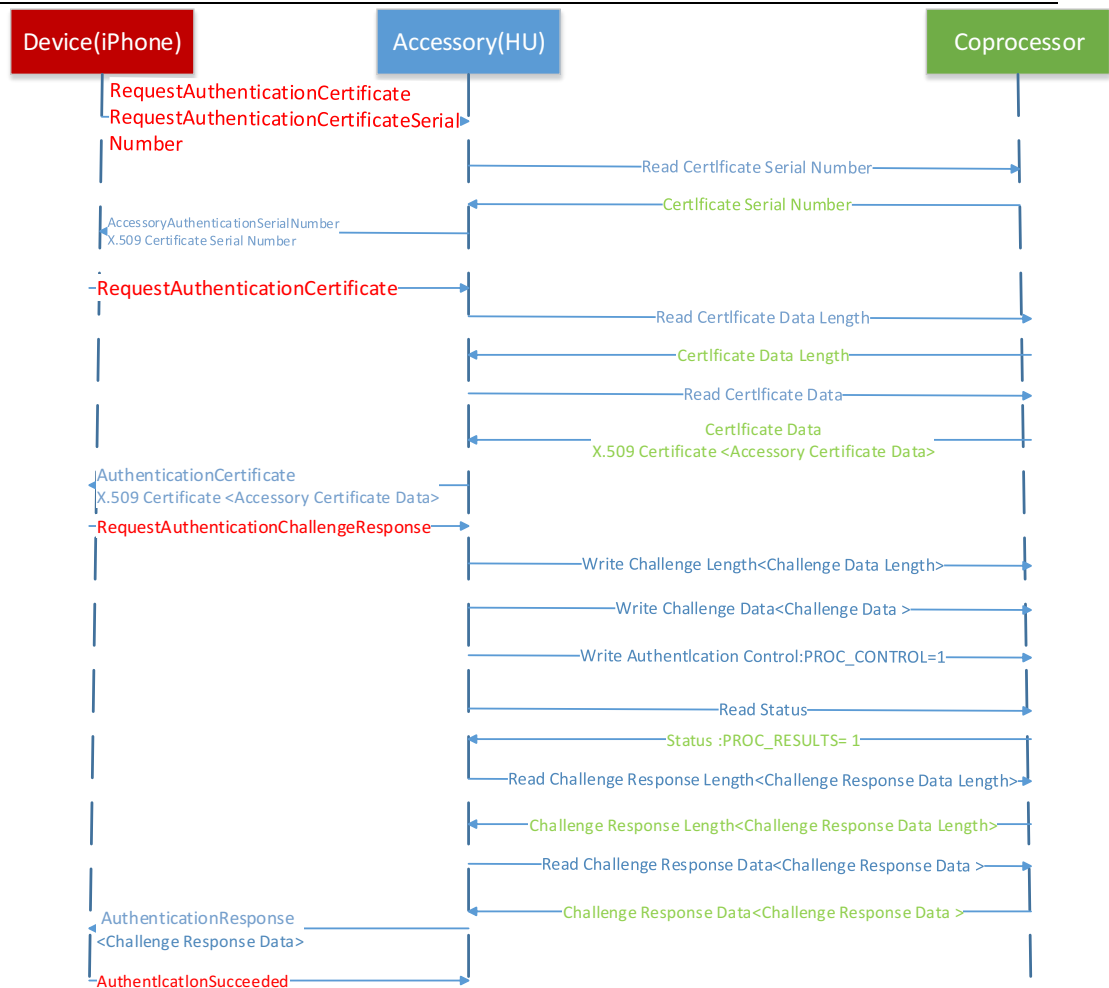


图 3-2: IAP2 认证时序图

6.1.2.3.2. Accessory Identification

经过 3.1 章节的设备鉴权，我们如果需要我们支持 EAP 的功能就需要做本节的功能认证，具体参考 MFI 认证文档的相关章节（R23 请参考 13 章 Accessory Identification 和 59 章 iAP2 Control Session Messages）。

一次典型的认证流程如下：

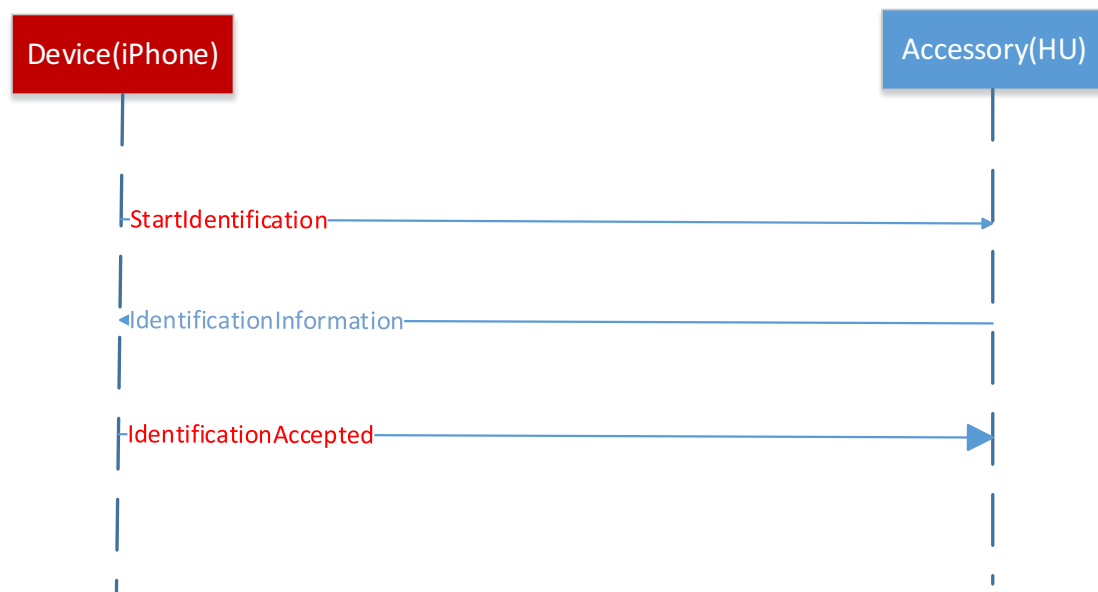


图 3-2: Accessory Identification 流程图

这里关键是 IdentificationInformation 命令所带的参数，具体含义请参考(R23 中 59.2.2IdentificationInformation)。其中 SupportedExternalAccessoryProtocol (R23 中的 Table 59-11) 及 USBHostTransportComponent (R23 中的 Table 59-15) 是非常重要的两个参数。SupportedExternalAccessoryProtocol 命令中的 ExternalAccessoryProtocolName 统一都是 com.baidu.CarLifeVehicleProtocol。

关于 Apple device 和 accessory 之间进行鉴权的过程，具体请依据苹果的 MFI 认证文档为准，R23 第 12 章(Accessory Authentication)、第 13 章 (Accessory Identification) 和第 59 章 (iAP2 Control Session Messages)。

6.1.2.4.拉起手机端 CarLife

拉起手机端的 CarLife 是使用苹果 MFI 认证协议里面的 App Launch 完成的。使用 App Launch 的命令绑定 CarLife 的 Bundle ID 即可将 iPhone 端的 CarLife 拉起。具体命令如下所示：

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

表 4-1: App launch 命令

LaunchAlert 的值如下表所示：

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.

表 4-2: AppLaunch Method enum

拉起主要应用场景如下表：

连接状态	手机 CarLife 运行状态	拉起命令
连接过程中	运行（在后台）	App launch
连接过程中	未运行	App launch
已连接	运行（在后台）	App launch

表 4-3：拉起的场景

百度建议使用带 Alert 的方式。因为不带 Alert 的方式需要 iPhone 未锁，并且停留在 Home 页面或者 Video/Music App 时才能拉起来，否则没有任何反应。带 Alert 的拉起方式则在任何情况下都会有弹框提醒。

6.1.2.5.EAP 连接消息结构

为了不对其他模块（比如音视频、语音、地图等模块）产生影响，尽量减少代码开发的工作量以及对其他连接方式的兼容，EAP 连接各个通道的消息结构保持不变，详见《CarLife 车机互联项目数据通信协议》。由于 EAP 实际上在传输的过程中只有一个通道，因此需要在原消息结构的基础上增加一个头用来标识该条消息是属于哪一个通道。具体结构如下：

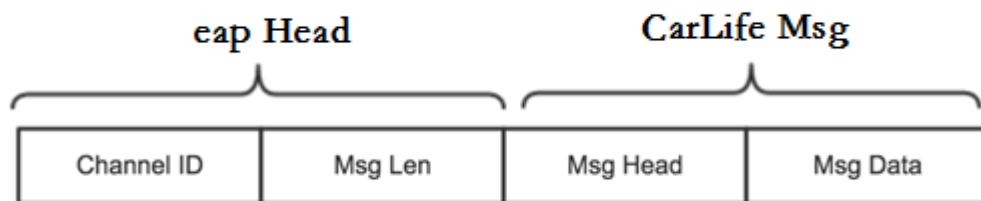


图 5-1：EAP 传输的消息结构

如上图，EAP Head 总占 8 个字节，Channel ID 和 Msg Len 各占 4 个字节，具体如下表：

字段名称	位偏移	大小(byte)	说明
Channel ID	0~3	4	标识该消息是属于哪个通道，Cmd, Video, Media, TTS, VR 以及 Touch 通道一次为 1, 2, 3, 4, 5, 6
Msg Len	4~7	4	后面 CarLife Msg 的整体长度
Msg Head	8~m+7	m	不同的通道长度不一样，Cmd 和 Touch 长度为 8 个字节，其他通道长度为 12 个字节
Msg Data	m+8~n	n-m-8	不同的通道数据类型不一样，Cmd 和 Touch 为序列化的 pb 结构，其他通道比如 Video 通道为视频流，

			Media 为 PCM 流
--	--	--	---------------

表 5-1:EAP 消息结构说明

6.1.2.6. 车机驱动层和车机端 CarLife 的交互协议

车机驱动层在完成 NCM 驱动加载、IAP2 认证过程中需要同步将其状态给车机端的 CarLife 应用层。这样做主要有两个目的：一是，CarLife 在连接的过程中需要根据这些状态来显示连接的进度。二是，当手机插入及拔掉时能够及时将这些状态反馈给 CarLife，便于 CarLife 拉起及做断开的处理。

6.1.2.6.1. 驱动和车机端 CarLife 的通信方式

根据车机系统的特性有多种通信方式（本地 socket、广播等），车机厂商可以根据不同的车机系统来选择。这里我们主要介绍一下本地 socket 通信。

1、通信的地址是/data/local/tmp/ncm.sock

2、socketName = "/data/local/tmp/ncm.sock";

1、车机端 CarLife 做客户端，车机端的驱动层（IAP2 认证程序）做服务端。

6.1.2.6.2. 车机驱动发送给车机端 CarLife 的消息

车机驱动在连接的不同时期给车机端 CarLife 发送不同的消息来告诉 CarLife USB 连接的具体进度。具体的命令及含义如下所示：

状态	值	适用的连接方式	备注
<i>sendUsbRoleSwitch</i>	0x01	EAP/NCM	车机驱动发送 RoleSwitch 命令的时候发送
<i>iap2AuthBefore</i>	0x02	EAP/NCM	进行 IAP2 认证前
<i>iap2AuthAfter</i>	0x03	EAP/NCM	IAP2 认证完后
<i>allocationNcmIp</i>	0x04	NCM	分配 IP 成功后
<i>eapFileReady</i>	0x05	EAP	EAP 设备端驱动映射的设备节点可以打开读写
<i>usbDisconnect</i>	0x10	EAP/NCM	Iphone 手机拔出后

表 6-1：车机驱动发送给 CarLife 的消息

6.1.2.6.3. 车机端 CarLife 发送给驱动的消息

车机端 CarLife 发送给车机驱动的消息只有拉起命令，车机端的 CarLife 本身是不能拉起手机端的 CarLife，只能依赖苹果的 MFI 认证协议里面的 App Launch 来完成。App Launch 的命令是在 IAP2 认证程序里面完成的，所以当需要拉起手机端 CarLife 时，我们需要发送一个

命令告诉 IAP2 认证程序，其收到这个命令后通过 IAP2 端口发送 App Launch 命令拉起手机端的 CarLife。具体命令如下：

状态	值	备注
pullUpCarLife	0x11	拉起 CarLife

表 6-2： CarLife 发送给车机驱动的消息

6.1.2.7.车机端 CarLife 增加对手机的心跳消息

EAP 连接时，不像网络 socket 的端口到端口的连接通信双方任意一方退出对方都能立刻知道。EAP 连接时，如果 USB 线保持连接，车机端 CarLife 退出的手机端感知不到。故需增加车机端到手机端的心跳机制。HU 在收到 MSG_CMD_PROTOCOL_VERSION_MATCH_STATUS 后，需要开始给 MD 发送心跳，每秒 1 次，心跳消息走视频通道，同之前 MD 给 HU 发送的心跳（MSG_VIDEO_HEARTBEAT = 0x00020002）。MD 在 EAP 连接方式下会通过心跳来判断 HU 的存活状态，若超过 3 秒没有收到心跳会主动断开连接。

6.1.2.8.车机端 CarLife 从 EA 通道读取数据的方法

关于 EAP 连接 HU 端的数据读取，有两点需要注意。

（1）HU 上层必须要有自己管理的数据缓冲区，不能依赖 EA 驱动的缓冲。EA 驱动收到 MD 发过来的数据之后，要立马全部读到 HU 上层自己的缓冲区里头。因为 EA 驱动的缓冲区有限，如果满了后，新来的数据就会冲掉之前的数据，导致数据错位，造成数据解析失败连接断开。

（2）HU 再 CarLife App 层启动一个专门的数据解析线程，从 HU 自己的缓冲区中读取数据并解析。操作缓冲区需要加锁，保证多线程安全。解析就按照我们 spec 定义的格式，先读 EAP HEAD，然后拿到 MsgLen，接着读取 MsgLen 长度的数据，如果当前缓冲区的数据不足 MsgLen，那就需要等待，等数据够 MsgLen 时再读取。然后解析 CarLife Msg，之后再循环读下一个消息。

6.1.2.9.车机端 CarLife EAP 连接动画百分比定义

车机端 CarLife 在连接中需要有连接动画（请参考 HMI 规范章节中的连接中），对于连接进度的百分比建议采用以下方式分配，便于用户反馈问题的定位：

百分比值（%）	备注
10	车机驱动发送 RoleSwitch 命令的时候发送
30	进行 IAP2 认证前
60	IAP2 认证完后
80	EAP 设备端驱动映射的设备节点可以打开读写
100	6 个 socket 连接成功

表 6-2： EAP 连接中的百分比

6.1.2.10. CarLife 与 CarPlay 的区别

CarLife 手机和车机采用的是 EAP 连接，而 CarPlay 是通过 NCM 端口来传输数据的另外他们在认证、驱动、连接上也存在一定差异，主要有如下几个方面：

1、认证

CarLife 做 MFI 认证需要通过 IAP2 的认证，CarPlay 另外还需要通过 CarPlay 的认证

2、驱动层

CarLife 和 CarPlay 在驱动层发送切换主从模式的命令存在差异，CarLife 的切换命令请参考 [6.1.3.2.2](#) 切换。CarPlay 的切换命令如下：

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

表 6.1-18: CarPlay 的端口切换命令

这里主要是 wValue 的值不同。CarLife 是 0: 仅仅是切换 NCM 端口；CarPlay 是 1: 在切换 NCM 端口的同时，让手机拉起 CarPlay。

3、连接

CarLife 的 EAP 连接是将一个 USB 通道虚拟成 6 个 socket 通道进行通信，而 CarPlay 是基于 IP 建立的 socket 连接，但是 CarLife 是建立的 TCPsocket 通道，而 CarPlay 是建立的 UDP socket 通道。另外 CarLife 同时支持 IPV4 和 IPV6 的连接，CarPlay 只能通过 IPV6 来建立连接。

6.1.2.11. CarLife 与 CarPlay 在车机端共存的拉起策略

Apple 在最新《Accessory Interface Specification》文档 23.2.4 CarPlay over USB 章节中，规定了一系列 CarPlay 通过 USB 连接需要自动拉起的规则：

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

考虑到 CarLife 与 CarPlay 在车机端的共存，在不违背 Apple 定义的规则的前提下，基于 iPhone 的互联我们有如下建议交互策略，目前已经有合作方车型采用该策略量产上市。

第一次连接 iPhone

- a) 对于 CarPlay

- 车机端需要弹出一个窗口，询问用户期望如何启动 CarPlay，“自动启动”或“手动启动”。
- 如果用户选择“自动启动”，那么本次将直接启动 CarPlay，以后每次用户连接 iPhone 都将自动拉起 CarPlay。
- 如果用户选择“手动启动”，那么将停留在车机端原页面，并且后续的每次连接将不会自动拉起 CarPlay，需用户在菜单中手动选择。

b) 对于 CarLife

- 当用户第一次在车机端菜单栏中选择了 CarLife 的按钮，车机端同样会弹出一个窗口，询问用户期望如何启动 CarLife，“自动启动”或“手动启动”。
- 如果用户选择“自动启动”，那么本次将直接启动 CarLife，以后每次用户连接 iPhone 都将自动拉起 CarLife。
- 如果用户选择“手动启动”，那么将停留在车机端原页面，并且后续的每次连接将不会自动拉起 CarLife，需用户在菜单中手动选择。

第二次连接 iPhone

- 车机端会判断之前用户是否已经执行了启动方式偏好选择。
- 若已经执行过选择，将按照用户偏好启动应用，不会再出现弹窗。
- 若之前未执行任何启动方式偏好选择，那么将再次弹出窗口询问。

修改启动偏好

- 车机端需要具备相关的设置选项，供用户设置或修改 CarPlay 及 CarLife 的启动偏好。

具体策略请查看如下流程图：

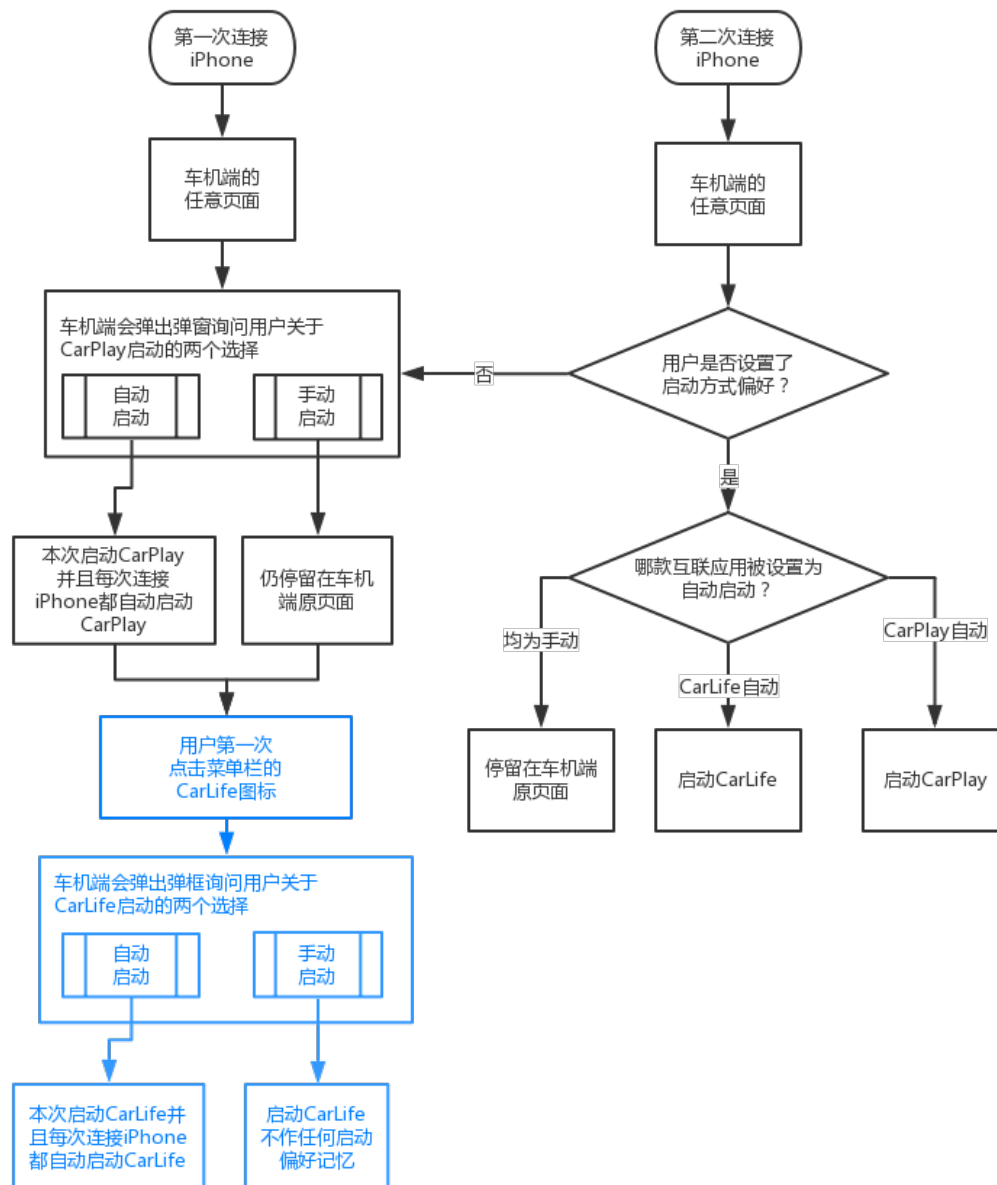


图 6.1-21: CarLife 与 CarPlay 车机端共存拉起策略流程图

6.1.2.12. CarLife MFi 认证流程

要支持 iPhone 的有线连接，车机厂商需要通过 MFi 认证。本章节主要介绍提交在 mfi.apple.com 上给车机提交 MFi 认证时需要注意的选项。

(1) Accessory Overview

技术类型，需要勾选 iAP

PFID 113308-0004	Accessory Status Active	Select Action ▼
Accessory Name DISPLAY AUDIO	Process Phase Production	
Base Model/SKU J16DAW0HA010	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) 固件和硬件

iAP 协议需要勾选 iAP2, 然后 USB 传输模式需要支持 USB Host Mode 和 USB Device Mode, iAP2 特性需要支持 App Launch 和 **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 Standard-A plug

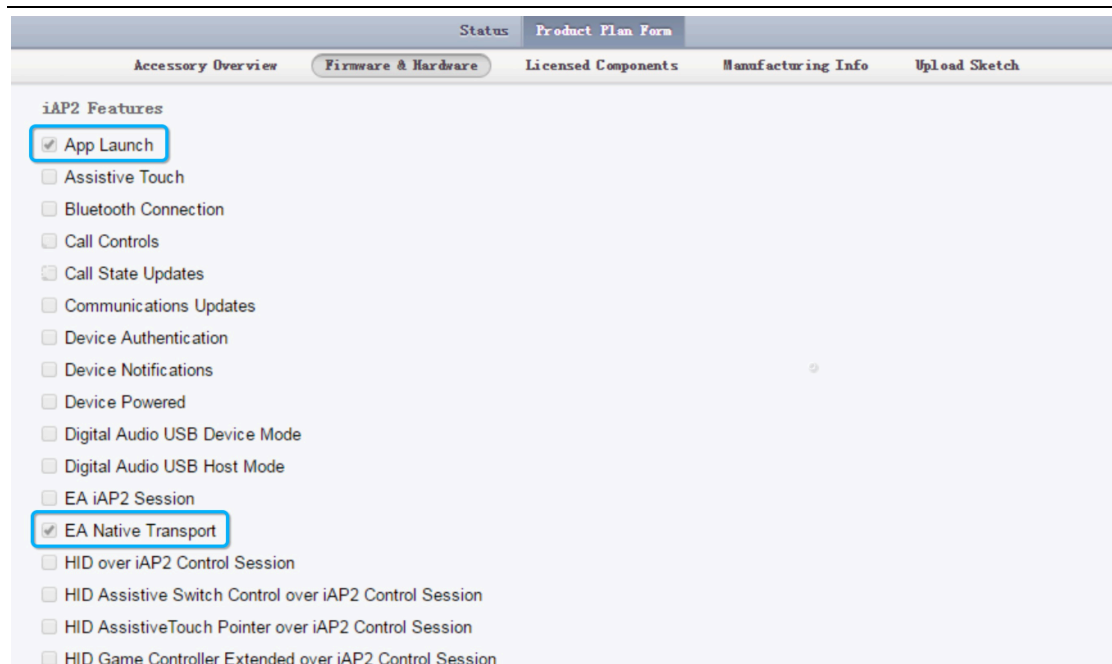
☐ USB Micro-A plug

☐ USB Standard-B receptacle or plug

☐ USB Mini-B receptacle or plug

☐ 3.5mm headset jack

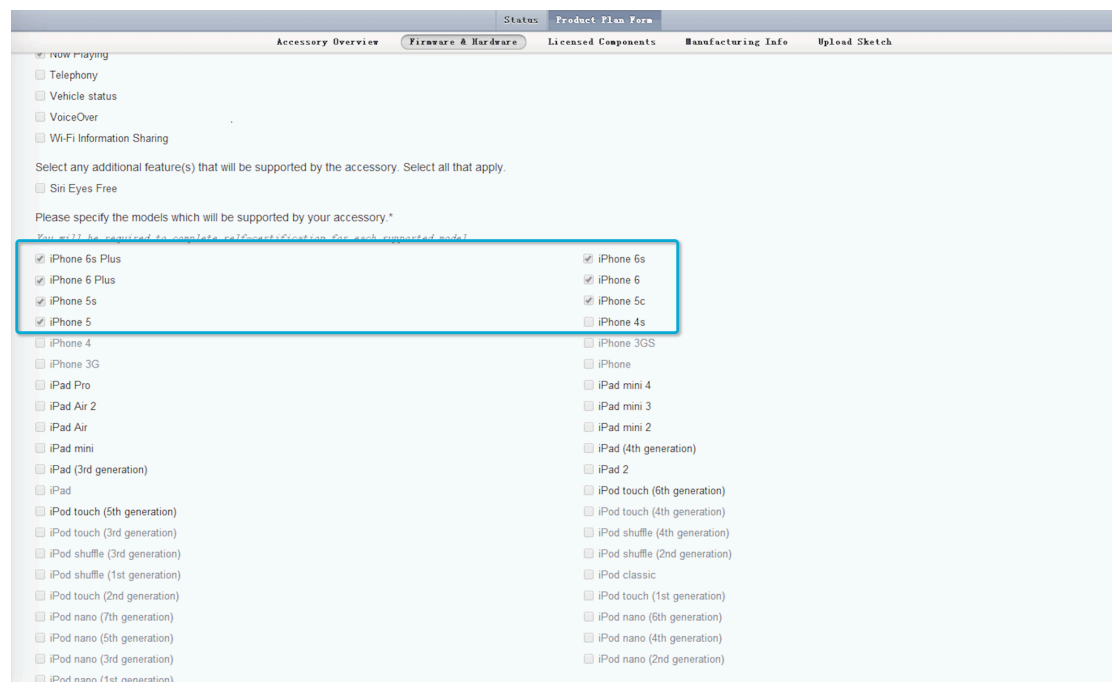
☐ Other non-USB



iAP2 Features

- ☒ App Launch
- ☐ Assistive Touch
- ☐ Bluetooth Connection
- ☐ Call Controls
- ☐ Call State Updates
- ☐ Communications Updates
- ☐ Device Authentication
- ☐ Device Notifications
- ☐ Device Powered
- ☐ Digital Audio USB Device Mode
- ☐ Digital Audio USB Host Mode
- ☐ EA iAP2 Session
- ☒ EA Native Transport
- ☐ HID over iAP2 Control Session
- ☐ HID Assistive Switch Control over iAP2 Control Session
- ☐ HID AssistiveTouch Pointer over iAP2 Control Session
- ☐ HID Game Controller Extended over iAP2 Control Session

需要支持的手机类型，iPhone5 以及以上机型。



Now Picking

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

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

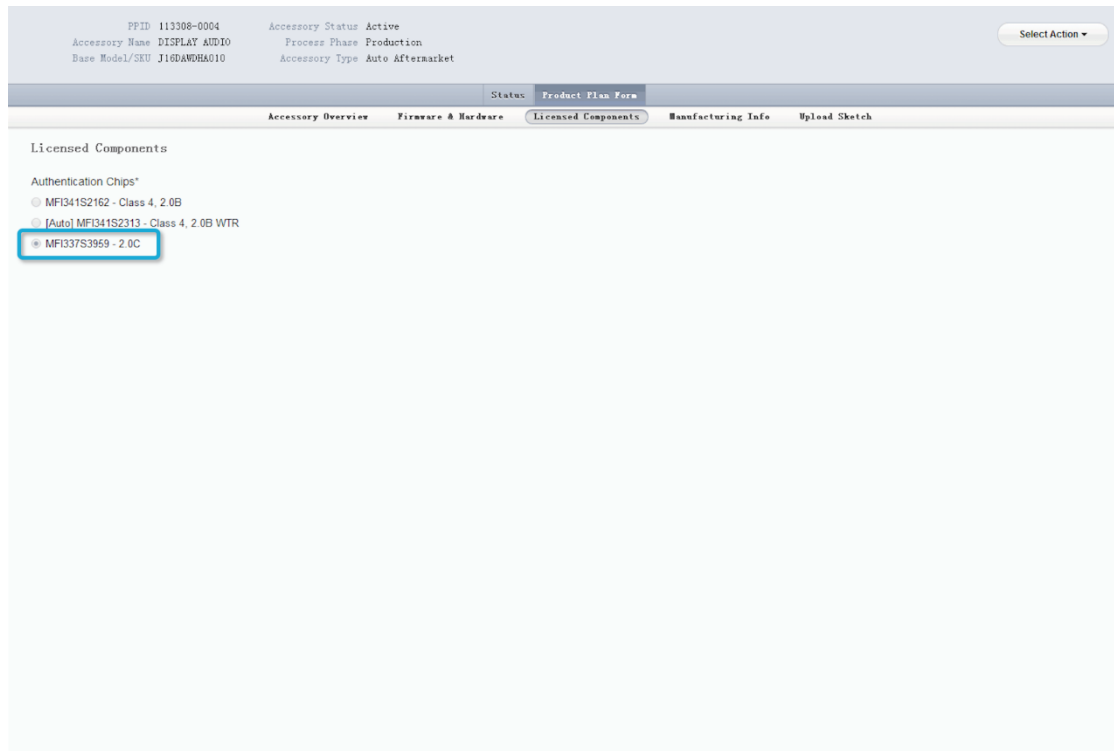
- ☐ Siri Eyes Free

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) 认证芯片

认证芯片选择 MFI337S3959-2.0C



(4) CarLife 的相关信息

CarLife bundleid: com.baidu.lbs.CarLife 这个 bundleid 也将用于 CarLife 的拉起。

6.2. 蓝牙自动匹配（车机端可选择性支持）

蓝牙自动匹配功能主要是指，车机和 Android 手机连接成功后，双方能够通过指定的协议完成交互，来完成蓝牙自动配对和建立连接的过程。在这个过程中，不需要用户手动的键入操作。并且只有当蓝牙成功连接后，蓝牙电话有关的功能才能生效。（IOS 系统目前不支持蓝牙自动匹配功能）

关于蓝牙自动匹配功能需要说明几点：

- 由于 IOS 系统的限制，IOS 系统目前不支持蓝牙自动匹配功能，需要用户手动去连接蓝牙
- 蓝牙自动匹配功能，需要车机端有一定的修改，主要是自动接受从手机发起的蓝牙连接请求
- 开启蓝牙自动匹配功能，需要在 MSG_CMD_HU_FEATURE_CONFIG_RESPONSE 中回应响应的选项。具体可以参考 MSG_CMD_HU_FEATURE_CONFIG_RESPONSE 消息的定义。
- 蓝牙自动匹配特性是选配功能，车机厂家可以根据实际情况来选择支持或者不支持。
- 如果车机之前连接过其他蓝牙设备，在开启自动匹配功能后，需要断开和之前设备的蓝牙连接，转而连接 CarLife 设备。
- 蓝牙自动匹配功能为 CarLife 的选配功能。车机厂商可以根据情况选择性支持。

6.2.1. 蓝牙配对模式

蓝牙的配对方式有多种，具体使用哪一种配对方式，依赖于蓝牙的版本以及设备的输入输出能力。在蓝牙 2.0 之前，传统的 PIN CODE 模式是唯一的选择。但是蓝牙 2.1 之后，引入了新的蓝牙配对方式叫做 Simple Secure Pairing。

目前 CarLife 的自动配对功能，主要适用于传统的 PIN CODE 配对和 SSP 中的 PASS KEY 比较配对方式。

6.2.1.1. PIN CODE 模式

在蓝牙 2.0 以及之前，这是唯一的配对支持的配对方式，配对的双方通过键入预先设置好的 PIN CODE 来实现配对。如果一方键入的 PIN CODE 不对，则配对失败。

6.2.1.2. SSP 模式

随着蓝牙 2.1 版本引入了 SSP 的配对模式，更对的配对方式可以被使用，目前 CarLife 主要支持基于 PASS KEY 比对的配对方式，其他的类似如 OOB, Pass Key Entry, Just Work 模式暂时不支持。

在 PASS KEY 比对的方式下，配对的双方只需要比较在两端出现 6 位数字 PASS KEY，来接受或者拒绝配对。如果数字相等，可以通过点击确认来接受配对，否则可以拒绝该配对。

6.2.2. 自动匹配流程

在开始自动配对之前一定要确保通过 MSG_CMD_HU_FEATURE_CONFIG_RESPONSE，告知手机支持蓝牙自动配对功能，如果手机端未收到车机的 FEATURE_CONFIG，则会导致配对不成功

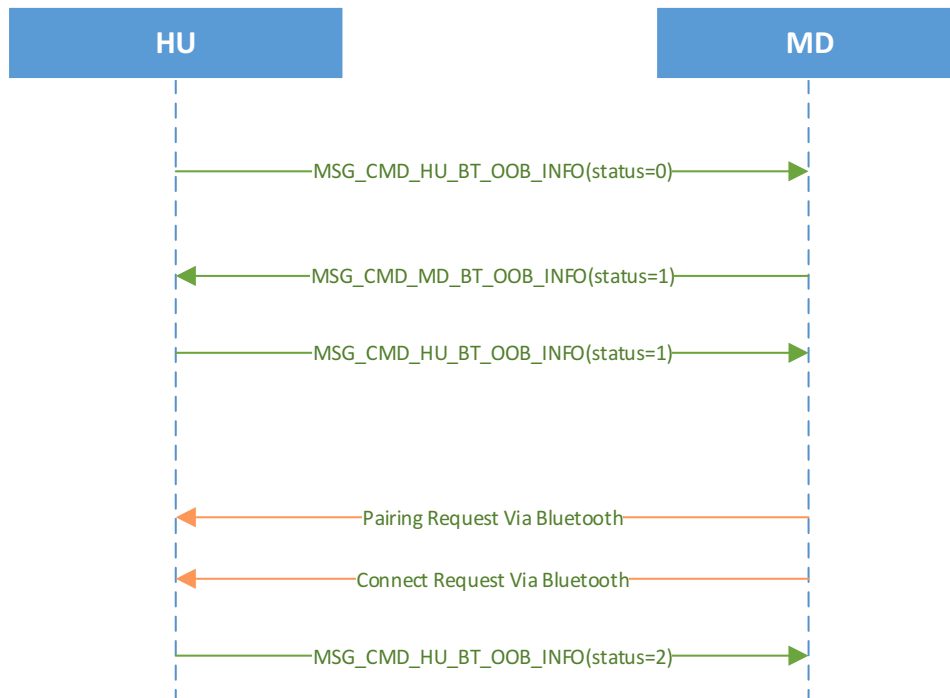


图 6.2-1

在 CarLife 连接上之后，车机会先发送 `MSG_CMD_BT_START_IDENTIFY_REQ` 给手机，手机收到该消息后，开始判断是否和这个车机已经建立 HFP 连接，如果已经连接上，则不需要开始自动匹配(IOS 不支持自动匹配)。如果没有连接上，并且是 Android 系统，则会发出 `MSG_CMD_START_BT_AUTOPAIR` 来通知车机开始自动匹配流程，**同时手机端开始进行蓝牙配对相关的操作**。因此，什么时候开始自动匹配流程，由手机来决定。

- 在收到手机发送的 `MSG_CMD_START_BT_AUTOPAIR` 命令之后，车机需要主动发送他的蓝牙 MAC 地址和 PIN CODE（如果使用 PIN CODE 模式）信息给手机。手机在完成一系列的初始化之后（包括自动使能蓝牙，检查蓝牙的连接状态），则需要进行到下一步。
- 手机发送它的蓝牙 MAC 地址，名字（可选）等信息给车机，告诉车机它的蓝牙已经在就绪状态（断开已经存在的连接），接下来车机也需要检测蓝牙是否连接，如果不是连接到该手机，车机也需要断开已经存在的链接，并处于就绪状态，等待接下来的配对请求。
- 车机端需要根据手机的地址，检查是否与之配对过，需要将是否配对的信息告诉手机，手机来选择是发起配对还是发起连接请求。
- 接下来手机发起配对请求或者连接请求，在配对和连接成功完成后，车机需要告诉手机连接完成。

更详细的流程可以参考下图：

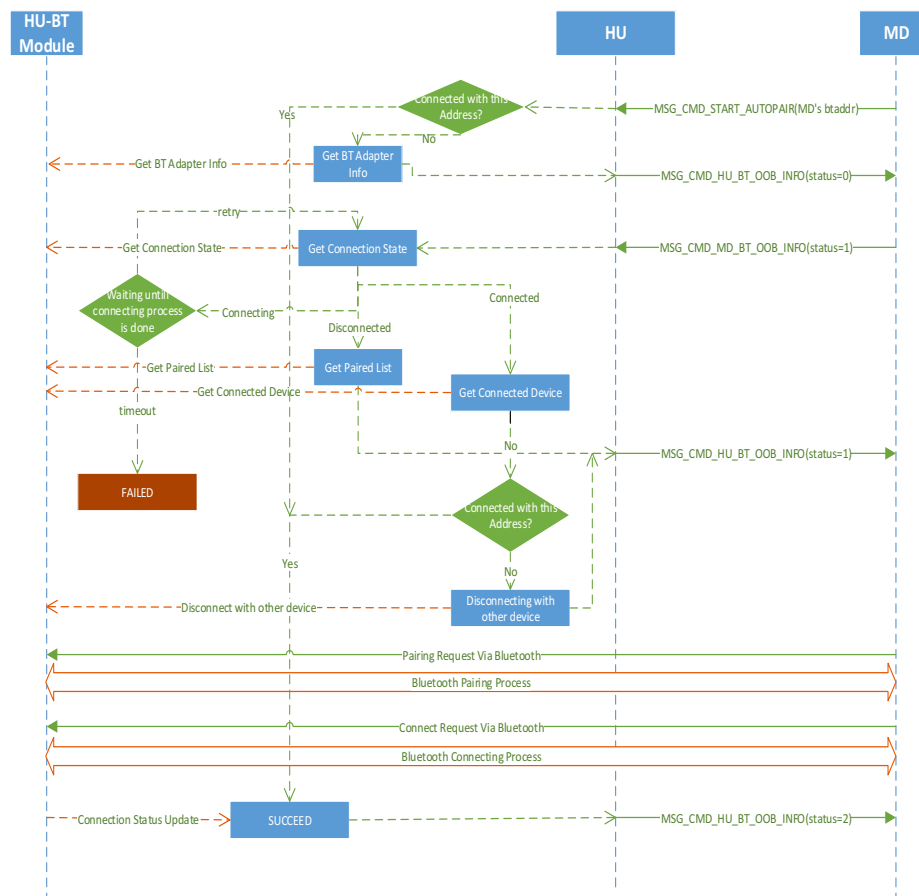


图 6.2-2

6.2.3. 初始化自动匹配内置蓝牙电话功能

在开始蓝牙自动匹配功能和内置蓝牙电话功能之前，需要完成相关初始化功能，主要包括：

- 告诉手机支持哪些特性（通过 MSG_CMD_HU_FEATURE_CONFIG_RESPONSE）
- 确定是否可以使用 CarLife 内置蓝牙电话功能。车机通过发送 START_IDENTIFY_REQ，手机收到后，回应 IDENTIFY_RESULT_IND 来告诉车机是否可以开始内置蓝牙电话功能。目前 Android 5.0 以下的版本，手机端不支持内置蓝牙电话功能。
- 在手机通过 IDENTIFY_RESULT_IND 消息告诉车机可以开始内置蓝牙电话功能后，车机需要屏蔽原生的蓝牙电话功能，由 CarLife 来接管电话操作。
- 在使用内置蓝牙电话功能之前，需要确保车机只连接一个蓝牙设备（部分车机支持连接多个蓝牙设备，在发送 START_IDENTIFY_REQ 之前，需要断开其他的蓝牙连接）

详细的流程见下图。

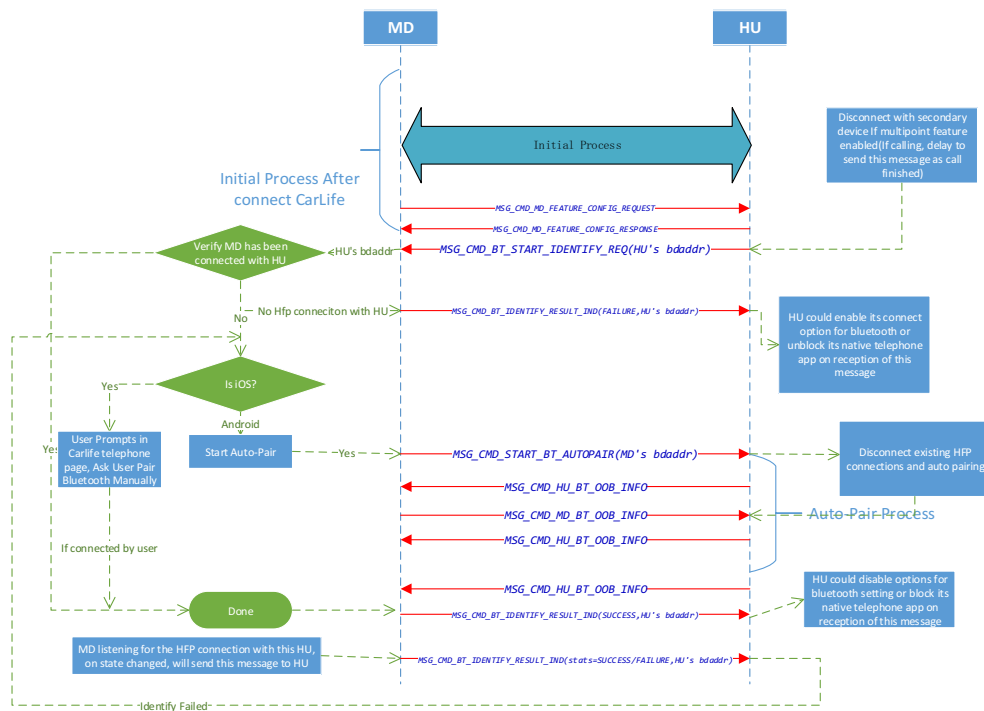
1. 当 CarLife USB 连接建立好之后，会开始手机和车机的握手过程，在这个过程中，需要车机通过 MSG_CMD_HU_INFO 消息来告诉车机的蓝牙地址给手机端。手机也会通过 MD_INFO

消息告诉车机手机自己的蓝牙地址（但是 iOS 系统由于自身的限制，无法获取自身的蓝牙地址，所以不会传）

2. 为了标定车机连上的蓝牙设备是不是 Carlife 设备，需要车机发送 `START_IDENTIFY_REQ`。
3. 手机端收到 `START_IDENTIFY_REQ` 后，检查是不是已经和这个车机建立了 HFP 连接，如果建立则不需要开始蓝牙自动匹配（Android），同时通过发送 `IDENTIFY_RESULT_IND` 告诉车机标定成功，可以开始蓝牙电话功能。
4. 如果没有连接上，同时手机端是 iOS 系统，则手机端将给出用户提示，需要用户手动去配对 iOS 手机和车机。如果是 Android 系统，则通知车机开始自动匹配的流程。
5. 车机收到开始自动匹配的命令后，需要断开已经存在的 HFP 连接，并开始自动匹配的流程
6. 手机将监听与车机的 HFP 连接状态，如果连接断开了，将会通过发送 `IDENTIFY_RESULT_IND` 来通知车机，同时中止 Carlife 蓝牙电话功能

（注：以上消息使用了简写, `MSG_CMD_BT_START_IDENTIFY_REQ = START_IDENTIFY_REQ`
`MSG_CMD_BT_IDENTIFY_RESULT_IN = IDENTIFY_RESULT_IND`）

具体流程如下图所示：



注：IDENTIFY FAILED 之后，ANDROID 和 IOS 只会提醒用户手动去连接蓝牙，而不会开启自动连接。蓝牙自动匹配过程只在 CARLIFE 连接上时发生一次，配对失败或者蓝牙主动断开连接, CarLife 不会再开启蓝牙自动连接。

6.2.4. 自动匹配接口定义

自动匹配过程中使用到的 PB 消息定义如下,主要用到了以下两个消息。

6.2.4.1.MSG_CMD_HU_BT_OOB_INFO

该消息从车机发送到手机，主要包含车机的蓝牙模组地址，以及 Pin Code 信息。具体消息中的字段定义如下表所示。

Field Name	Type	Optional/ Required	Description
Address	String	Required	48 位的车机蓝牙地址
Passkey	String	Optional	如果使用 Passkey 比对模式，该字段不填。如果使用 Pin Code 模式，需要填写使用的 Pin Code
Hash	String	Optional	保留给 OOB 模式字段
randomizer	String	Optional	保留给 OOB 模式字段
Uuid	String	Required	未使用字段
Name	String	Required	车机蓝牙的名字
Status	Integer	Required	用来指示蓝牙自动配对过程的各种状态阶段，主要的状态有： IDLE = 0: 初始化状态 READY = 1: 就绪状态，主要是指蓝牙模组断开已经存在的连接，准备好响应新的连接请求。 CONNECTED = 2: 连接成功状态，主要指车机已经成功连上了目标手机设备。
ispaired	Integer	Optional	用来指示，车机是否已经和目标手机设备配对过。

表 6.1-20

6.2.4.2.MSG_CMD_MD_BT_OOB_INFO

该消息从手机发送到车机，主要告诉车机关于手机的蓝牙地址，状态信息。具体消息中的字段定义如下表所示。

Field Name	Type	Optional/ Required	Description
btAddress	String	Required	手机的蓝牙地址
passKey	String	Optional	未使用字段
hash	String	Optional	保留给 OOB 模式使用字段
randomizer	String	Optional	保留给 OOB 模式使用字段
uuid	String	Optional	未使用字段
name	String	Optional	手机的蓝牙名字
status	Integer	Required	指示蓝牙自动配对过程的各个状态阶段，具体状态定义可以参考表 8.2-1

表 6.1-21

6.2.4.3.MSG_CMD_START_BT_AUTOPAIR

该消息从手机发送到车机，用来通知车机开始蓝牙自动匹配的流程，同时断开现有的 HFP 连接。如果是 iOS 系统,手机端则不会发送该消息。

具体消息定义如下：

Field Name	Type	Optional/ Required	Description
ostype	Integer	Required	手机的操作系统类型： 0: Android 1: iOS (Not Used)
address	String	Optional	如果是 Android 系统，则需要填写 MD 的蓝牙地址，如果是 iOS 系统则不需要填写

7. 视频引擎

7.1. ~~bdsc~~ 的处理方式

~~CarLife 在 Android 5.0 或更高版本不需要 bdsc 的支持，仅在低于 Android 5.0 版本的 Android 系统移动设备和 HU 连接后，需要由车机推送一个由百度提供的，名为 bdsc 的可执行文件至 MD，并使其在手机端运行起来，以便完成 CarLife 视频相关的功能。~~

~~根据移动设备不同 Android 版本，bdsc 也有不同的版本与其对应，下表为 bdsc 与 Android 版本的对应关系。~~

MD 的 Android 系统版本	对应的 bdsc 文件名称
Android 4.1	bdsc16
Android 4.2	bdsc17
Android 4.3	bdsc18
Android 4.4.2	bdsc19
Android 4.4.4	bdsc19_01

~~物理连接建立后，由车机使用 adb 命令，向手机推送对应版本的 bdscXX，获取 Android 手机端系统版本等信息的操作步骤，详见[连接引擎](#)，使用 bdsc 的相关命令如下。~~

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

~~为了确保之前没有残留的 bdsc 进程，需要先查看手机的进程列表，并 kill 掉 bdsc 进程：~~

~~adb shell ps~~

~~adb shell kill 9 [残留 bdsc 的进程号]~~

~~之后执行 adb 命令使其运行。~~

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

~~adb shell /data/local/tmp/bdsc~~

~~在 bdsc 进程成功运行之后，才能发送视频传输开始的消息，完成后续连接过程。~~

7.2. 动态帧率机制

CarLife 中的动态分辨率机制分为两个部分，即 HU 侧的动态分辨率调整和 MD 侧的动态分辨率调整。

7.2.1. 车机侧的帧率调整

对于解码速度不能达到 30 帧每秒的 HU 设备，或运行时某一段时间解码能力下降的 HU 设备，CarLife 通信协议中提供了帧率调整的协议，用于控制 MD 侧视频帧的产生最大速度。

HU 侧发送的帧率调整参数，仅在不小于 3，不大于 30 时生效。

具体协议内容及用法，详见通信协议的 MSG_CMD_VIDEO_ENCODER_FRAME_RATE_CHANGE 消息：[相关消息体](#)。

7.2.2. 移动设备侧的帧率调整

MD 侧视频帧产生速度的最大值，不应超过 HU 侧发送的帧率调整参数，依据 MD 侧的帧率调整策略，在合理范围内浮动。

7.3. 动态分辨率机制

CarLife 中的动态分辨率机制旨在适配不同的 MD 设备。

MD 与 HU 建立通信连接后，HU 向 MD 侧发送希望获取的视频分辨率参数等信息；MD 接收到信息后，根据自身设备的特性，计算出自己所能支持的视频分辨率参数，反馈给 HU 侧。由于 MD 编码模块条件的限制，不能完全匹配 HU 的宽度参数，也难以完全匹配自身的显示区域的宽高比例，最终的参数会是一组尽量靠近的宽高值参数对。相关消息体的使用时序，详见[建立连接时视频消息相关时序图](#)。

MD 和 HU 双方经过上述的消息交互后，即可使用 MD 侧最终反馈回的视频参数，对视频数据通道中的视频数据进行解码和显示。

若最终反馈回的参数与 HU 侧发送的参数不相同，HU 侧应依据自身策略，对显示的图像进行适当的缩放和拉伸处理。

7.4. 超宽或者超高屏幕适配策略

HU 和 MD 在建立连接的初始阶段，会通过 CarLife 协议字段商定视频分辨率；MD 侧会根据 HU 传递过来的宽度参数，结合 MD 侧 CarLife 的显示区域比例，计算出视频的高度参数。

对应超宽或者超高的屏幕，宽高比跟 MD 侧屏幕宽高比差距太大，全屏显示会导致拉伸或者压缩明显，显示异常。建议 HU 端设定一个合理的显示区域，对应超宽（超高）的屏幕可以左右（上下）留黑边或者显示特定的信息。然后，HU 把显示区域的宽高参数传给 MD 侧，之后在显示时，由 HU 侧使用合适的缩放策略，对视频图像的缩放显示，以填充 HU 侧划定的显示区域。

对于超宽的屏幕，上下填满，左右对称留黑边。



类似，对于超高的屏幕，则左右填满，上下对称留黑边。



如果黑边的区域比较大，可以显示一些车厂自定义的信息，比如车机原生的系统主菜单、或者显示日期，时钟等等。

8. 音频引擎

8.1. CarLife 音频引擎架构

CarLife 中的音频模块主要包括导航 TTS 音频模块（Navi TTS Module）、媒体模块（Media Module）和语音助手模块（VR Module），如图 10-1 所示，

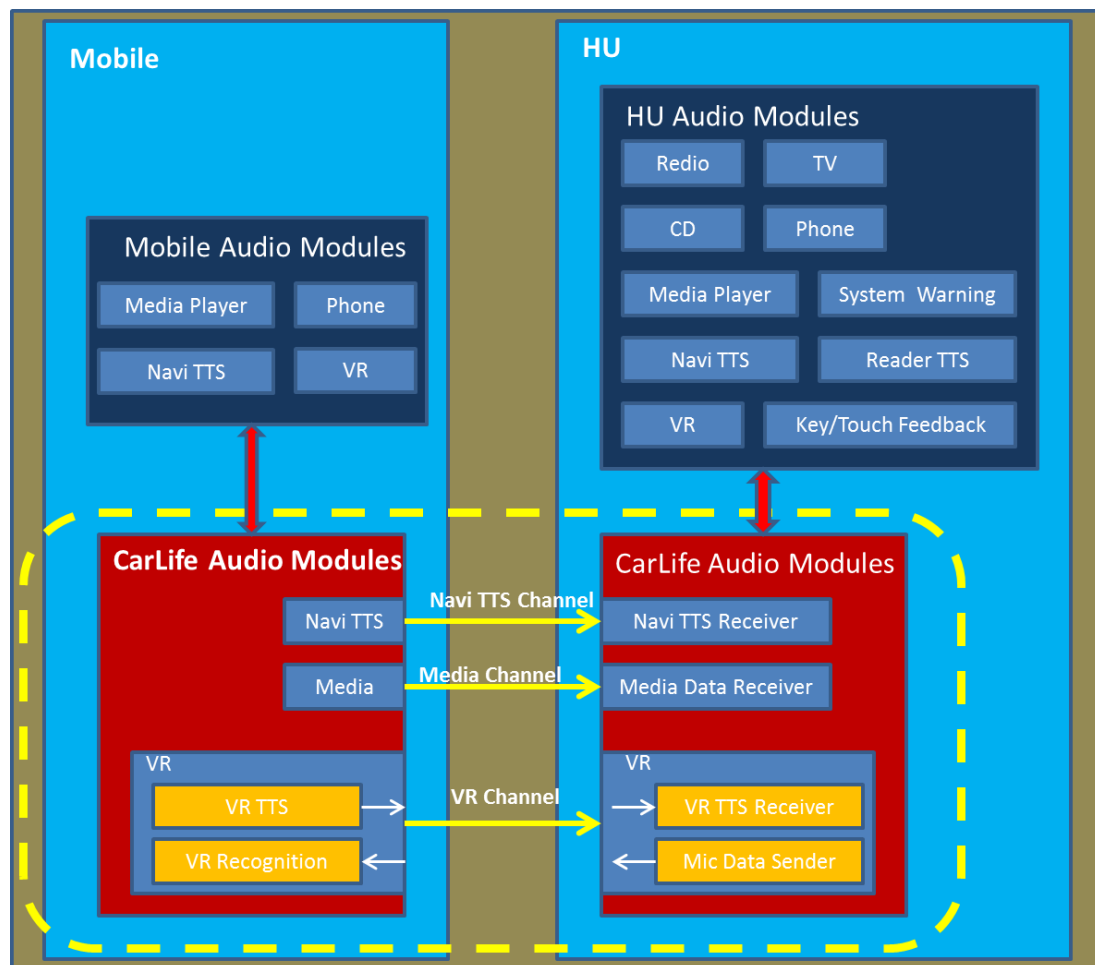


图 8.1-1

其中，

导航 TTS 音频模块通过 Navi TTS 通道将需要播放的音频数据流以 PCM 的形式发送到车机端，供车机端对应模块处理播放。

媒体模块通过 Media 通道将需要播放的音频数据以 PCM 流形式发送到车机端，供车机端对应模块处理播放。

语音助手模块一方面将需要播放的语音提示音通过 VR 通道发送到车机端，供车机端对应模块处理播放；同时，语音助手模块通过 VR 通道接收来自车机端的 MIC 录音，从而实现语音识别。

8.2. 音频流类型

CarLife 中音频流类型一共有四种，使用三个独立的通道进行传输（其中，VR TTS 与 VR MIC 使用同一通道，但数据流传输放向相反），具体的相关参数及说明如表 10-1 所示：

音频流类型	音频流格式	使用通道	类型描述
Media	编码方式：PCM； 采样率：取决于手机端需要播放的音乐文件（一般为44.1kHz）； 通道数：取决于手机端需要播放的音乐文件（一般为双声道）； 量化格式：16bits（小端序）；	Media 通道 （手机->车机）	手机端将压缩的音文件（如 MP3、AAC）解码为 PCM 格式数据流，传输给车机进行播放
Navi TTS	编码方式：PCM； 采样率：16kHz； 通道数：单声道； 量化格式：16bits（小端序）；	Navi TTS 通道 （手机->车机）	手机端将导航提示音（TTS）发送到车机端进行播放
VR TTS	编码格式：PCM； 采样率：16kHz； 通道数：单声道； 量化格式：16bits（小端序）；	VR 通道 （手机->车机）	手机端将语音提示音（TTS）发送到车机端进行播放
VR MIC	编码格式：PCM； 采样率：16kHz； 通道数：单声道； 量化格式：bits（小端序）；	VR 通道 （车机->手机）	车机端将 MIC 录音传输到手机端，供 VR 的引擎进行解析

表 8.2-1

8.3. 音频焦点类型

音频焦点是指，需要播放声音的音频模块（如音乐、导航提示音）对发声单

元（speaker）的占有权。音频模块播放声音前，要先申请音频焦点，音频焦点申请成功，表示当前音频模块占用发声单元，可以播放模块声音；音频焦点申请失败，则不能进行播放；

CarLife 中音频焦点类型一共有三种：长期独占型、短期独占型以及可混音型。

其中，

长期独占型：当申请该类型音频焦点时，CarLife 通知系统，表示其音频应用将长时间占用音频播放设备。该类型音频焦点主要用于 Media 类型音频内容的播放，比如本地音乐、乐播、网易云音乐以及第三方音频 SDK 类应用，如喜马拉雅 FM。该类型音频焦点在申请过后，不会主动释放。只有被其他音频应用抢占时，才会失去自身的音频焦点。

短期独占型：当申请该类型音频焦点时，CarLife 通知系统，表示其音频应用只是在较短的时间内占用音频播放设备，使用完之后马上释放播放设备，并通知系统对音频播放设备的占用已结束。该类型的音频焦点主要用于 VR 的工作期间。当 VR 启动时申请该类型的音频焦点，当 VR 行为结束时释放该类型音频焦点。

可混音型：当申请该类型音频焦点时，CarLife 通知系统，表示其音频应用将短期占用音频设备。但在其占用音频设备时，其他音频类应用可以同时使用该音频播放设备，但必须降低自身的音量，保证申请可混音型的音频应用在前台播放。该类型的音频焦点主要用于导航 TTS 的播放，导航播放时，导航的 TTS 播放音需要在前台进行播放，其他音频应用需要降低自身音量在后台混音播放。当导航 TTS 播放时，申请该类型的音频焦点，当导航 TTS 播放结束时释放该类型的音频焦点。

CarLife 中音频应用与音频焦点申请类型的对应关系如表 10-2 所示：

音频模块	音频焦点类型
Media	长期独占型
Navi TTS	可混音型
VR	短期独占型

表 8.3-1

8.4. 音频焦点管理策略

CarLife 中的音频模块在同系统中其他音频应用在使用时，遵从如下的音频焦点使用规范：

Media 类音频应用

- （1）焦点申请：Media 类型的 CarLife 音频模块（如本地音乐）在播放前申请长期独占型音

频焦点；焦点申请失败（如正在接听电话），Media 类型的音频模块需暂停；

（2）**焦点释放：**Media 类型的 CarLife 音频模块在播放结束后不主动释放音频焦点。只有被其他类型音频应用抢占（如其他音乐播放器进行播放）才会失去其所占有的音频焦点；

（3）**焦点监听：**Media 类型的 CarLife 音频模块监听到被长期独占型（其他音乐播放器）或短期独占型音频应用（VR）抢占焦点时，该模块做暂停播放处理；当重新获得音频焦点时（VR 动作结束），继续播放暂停的音乐；当监听到可混音型音频应用（Navi TTS）抢占焦点时，该模块降低自身音量，同抢占焦点的应用进行混音；

Navi TTS 音频应用

- （1）**焦点申请：**Navi TTS 在播放前申请可混音型音频焦点；焦点申请失败（如正在接听电话），则停止进行 TTS 提示播放；
- （2）**焦点释放：**当 Navi TTS 提示音播放完毕后，主动释放所占有的音频焦点；
- （3）**焦点监听：**Navi TTS 模块监听到被短期独占型音频应用抢占焦点时（如启动 VR），Navi TTS 不播放提示音；

VR TTS 音频应用

- （1）**焦点申请：**VR TTS 在播放前申请短期独占型音频焦点；焦点申请失败（如正在接听电话），则停止进行 VR TTS 提示音播放；
- （2）**焦点释放：**当 VR TTS 提示音播放完毕后，主动释放所占有的音频焦点；

焦点监听：VR TTS 模块监听到被短期独占型音频应用（如电话接入）抢占焦点时，停止播放提示音；

8.4.1. duck 处理

当 CarLife 中的 Media 模块和导航模块同时工作时，Media 模块的音频数据和导航模块中的 TTS 提示音会分别通过 Media 通道和 Navi TTS 通道同时传输到车机端。车机端结合自身的音频处理方式，将 Media PCM 数据和 Navi TTS PCM 数据混音后进行播放。其中，Media 声音需要降低音量，保证 Navi TTS 的声音能够清晰的被驾驶员听到。Navi TTS 与 Media 音频同时播放时，处理策略如图 10-2 所示：

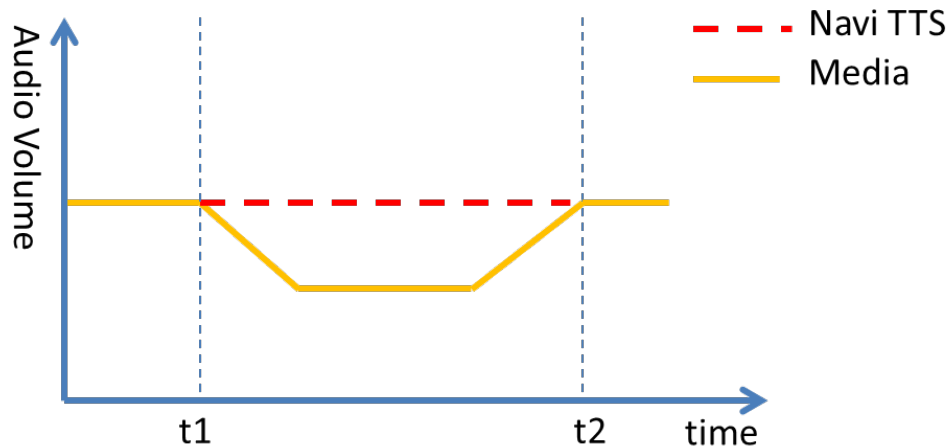


图 8.4-1: Navi TTS 与 Media 同时播放处理策略

其中，t1 时刻 Navi TTS 开始播放，t2 时刻 Navi TTS 停止播放。

8.4.2. mute 机制

车机端 CarLife 在使用过程中，车机端在高优先级音频应用需要独占发声设备的场景下需要停止 CarLife 中音频数据的播放，停止方式主要有两种：

1. 车机端接收到来自手机端的 Navi TTS、VR TTS、Media 数据，可以直接将接收到的数据进行丢弃，不进行播放；该情况下，手机端不会停止音频数据的发送。
2. 车机端可以通过命令通道的控制命令 MSG_CMD_MODULE_CONTROL，停止手机端 CarLife 的音频模块（比如导航、语音助手、音乐。其中，音乐模块除了可以暂停外，还能通过该命令恢复部分）。手机端播放暂停后，会将模块状态的变化信息通过命令通道的消息 MSG_CMD_MODULE_STATUS 同步给车机端。
3. 针对电话过程中的 mute 机制，指的是通电话过程中，mute 掉车机本地的 mic，使得电话远端听不到近端的声音，主要分为以下两种情况：

1) 车机端已经集成了 CarLife 蓝牙电话功能：

整个电话过程将使用 CarLife 提供的内置电话界面来进行操作，第一，用户此时可以点击界面内的 Mute 按键，通过在 USB 通道上发送蓝牙电话中定义的 mute mic 命令（见章节 11.3.2.1），来禁用本地 MIC。

第二，用户可以按下车机上的 Hard Key 来禁用本地 MIC，在这种情况下，车机来负责禁用 MIC，但是需要在按下该按键时，将相应的 KEY CODE 发送给手机端，从而让手机端更新界面的显示。

2) 车机使用 Native 的蓝牙电话 APP：

此种情况下，通话过程使用的都是车机 Native 的电话界面，用户 Mute Mic 可以通过点击界面内的 Mute 按键或者车机自身的 Hard Key。整个状态的处理由车机自身来处理

8.5. 与车机本地音频的冲突处理策略

车机端的音频应用同 CarLife 中的音频应用在使用时，需要遵从如下的音频焦点使用规范：

1. 在进行音频播放前需先进行音频焦点申请；
2. 在音频播放结束后需根据实际情况决定是否释放所持有的音频焦点；

对于长期独占型音频焦点（如音乐播放），在结束播放单元占用后，不需要主动的释放音频焦点；

对于可混音型音频焦点（如导航提示音播放），在结束播放单元占用后，需要立即主动释放音频焦点；

对于短期独占型音频焦点（如语音助手），在运行结束时，需要立即主动释放音频焦点；

3. 在播放音频时，同时需要对 CarLife 音频应用的焦点使用进行监听，并做相应的处理；

8.6. 音频应用优先级

音频焦点为系统中的不同音频应用提供了状态同步的机制，但该机制不能完全满足所有的音频使用场景。所以，在遵从音频焦点机制的同时，也需要考虑各个音频模块的优先级策略。在使用 CarLife 的音频系统中，各个音频模块的优先级关系为：

电话>VR>Media/导航

其中，电话的优先级最高，Media 和导航的 TTS 提示音最低（FM、蓝牙音乐、本地音乐均为 Media 类音频模块）。

当电话模块工作时，其他音频类应用不能播放声音；当 VR 类应用启动工作过程中，Media/Navi TTS 类音频应用不能播放声音。

8.7. VR 功能

CarLife-VR 功能是基于百度语音技术部提供的 SDK 来开发的，基础能力包括语音识别和语音唤醒，其中语音识别是在/离线识别融合，唤醒是离线识别。

目前只支持中文指令集。

百度语音引擎对输入语料的要求有六点：

- 1) 16K采样率
- 2) 16bit位深（小端序、有符号数值）

- 3) 单通道
- 4) 幅值在5000左右
- 5) 信噪比在 12db 以上

鉴于以上五点要求，合作方需要对车机 MIC 录制到的语料进行评估，评估细则请参考《CarLife 录音语料评测简易教程》。

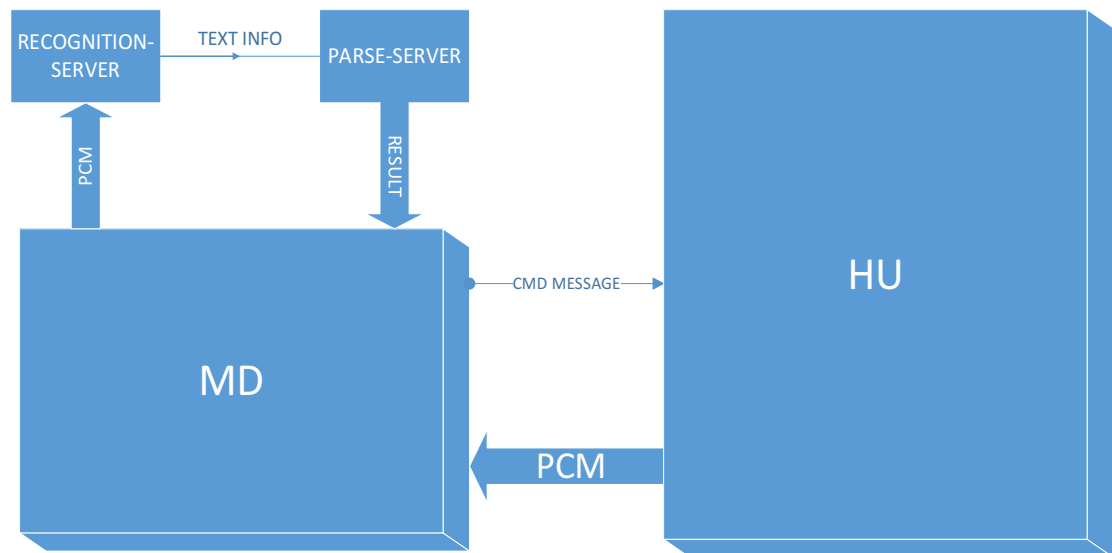


图 8.7-1: 语音结构框图

如图表 1 所示，单机状态，CarLife-VR 采用手机自带 MIC 录音；联机状态，VR 模块默认采用车机端 MIC 录音，并将 PCM 流数据实时地转发给语音服务器，并等待语音服务器返回识别结果。具体实现方法请参考《[12.6 语音消息](#)》

若是车机端 MIC 无法达到语音引擎的录音要求，CarLife 也支持 OEM 厂商选择手机自带 MIC 录音。具体实现方法请参考《[12.6 语音消息](#)》

初次启动 CarLife 时，语音唤醒默认关闭，CarLife 设置界面里有唤醒功能的开关。

CarLife 的语音指令会持续的新增，完善和优化，这部分功能只依赖于手机端，请以最新手机端支持的指令为准。

以下为部分 CarLife 所支持的语音指令集：

前往	前往首页
	前往电话
	前往地图
	前往音乐
音乐	播放音乐

		上一首
		下一首
		播放
		我要听 XXX (人名) 的歌
		我要听 XXX (歌名)
	地图	回家
		去公司/回公司
		附近的 XXX (加油站等)
		导航去 XXX
		我要去 XXX
	电话	打电话给 XXX (人名)
		打电话给 XXX (电话号码)
	帮助指令	帮助 (展开语音帮助页)
		打开小度语音唤醒/打开语音唤醒
		关闭小度语音唤醒/关闭语音唤醒
		检查更新 (Android)

表 8.7-1

9. 蓝牙电话

9.1. 前言

为了提高 CarLife 在拨打/接听电话过程中的体验，由于 Android,iOS 系统本身的限制，很多电话相关的事件和动作，CarLife APP 无法获得，因此需要作为蓝牙免提设备（Hand-Free Device）的车机来帮助完成。

同时由于一般的车机本身系统里面自带默认的蓝牙电话 APP，因此需要和车机厂商合作完成相关功能

蓝牙电话功能能否启用的前提是，需要车机和 CarLife 手机的蓝牙 HFP 已经连接成功，如果没有，内置的蓝牙电话功能将不会被启用。

目前，针对 Android 系统，只有 5.0 以上支持 CarLife 内置蓝牙电话功能，而 IOS 系统全部支持内置蓝牙电话功能。

CarLife 内置蓝牙电话功能能够极大的提高，用户在打电话的过程中使用 CarLife 的体验，强烈建议厂家支持该功能。

关于接入 CarLife 内置蓝牙电话功能，常见的问题如下：

- IOS 系统手机，Android AOA 连接方式的手机都支持内置蓝牙电话功能。
- 在使用内置蓝牙电话功能时，要求车机在连上 CarLife 之后，只能连接一个蓝牙设备，不支持多个连接
- 目前 CarLife 内置蓝牙电话不支持多方通话，如果是多方通话，还是由车载原生蓝牙电话处理。
- 车机在收到 MSG_CMD_BT_IDENTIFY_RESULT_IND 后，如果结果为 TRUE,则说明可以开始使用内置蓝牙电话功能，这时需要屏蔽车机原生的蓝牙电话界面。如果为 False 则说明内置蓝牙电话功能没有启用。

在开始 CarLife 内置蓝牙电话功能之前，需要完成初始化流程，请参考章节 6.2.3

9.2. 接口框架

9.2.1. 整体结构

如下图所示，由于在 iOS, Android 系统上面，CarLife 和 Telephone 之间的接口不完善，比如在 Android 没有提供接听电话给第三方 APP. 为了能够在 CarLife 应用内完成对电话的操作，因此需要通过先将消息发给车机，由车机来完成电话的有关操作。

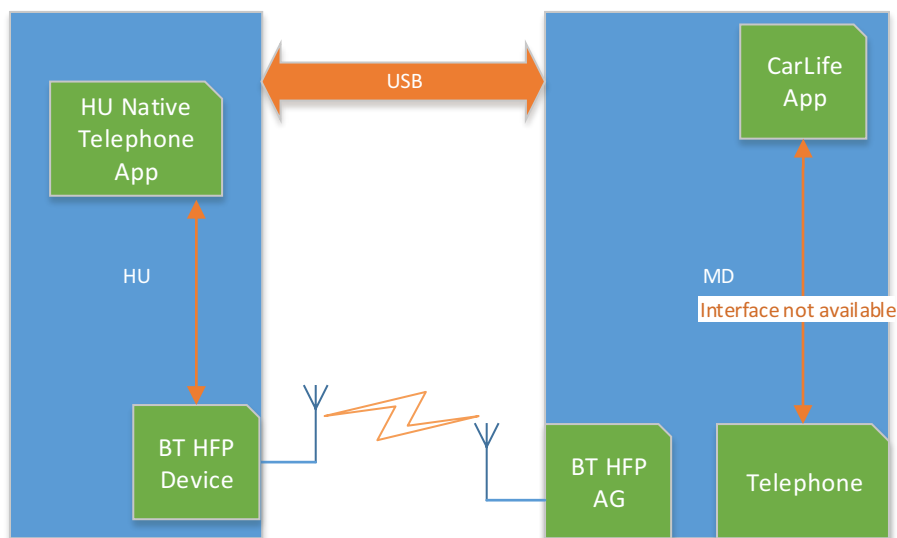


图 9.2-1

9.2.2. 蓝牙电话消息序列

一个典型的接听来电的场景，消息流如下图所示（由于 Android iOS 系统的限制，CarLife APP 内无法直接接听来电，所以需要借助车机蓝牙作为 HFP Client 的能力来接听电话）

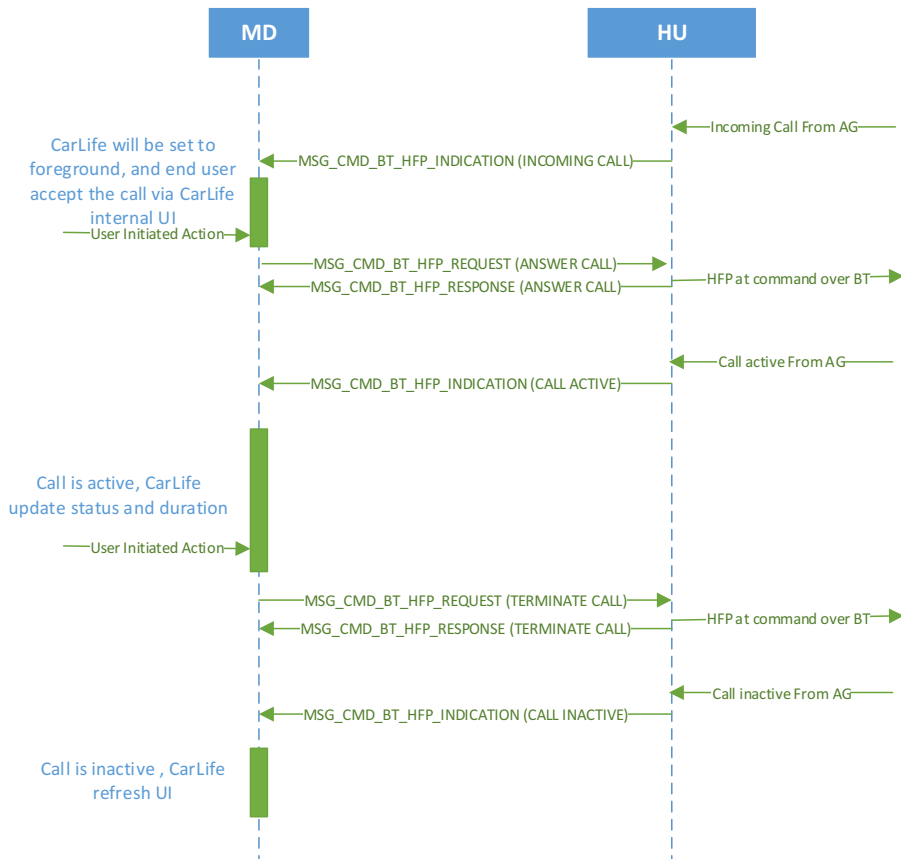


图 9.2-2

当车机的蓝牙和手机的蓝牙已经连接成功后，一个典型的来电处理流程需要经过以下几个步骤：

- 车机蓝牙在收到新的来电提示后，通过预先定义的 PB 消息，通知 CarLife 来电的状态。
- CarLife APP 取代车机原生的蓝牙电话界面，再前台弹出新的来电提醒。
- 用户点击接听或者拒接按键，后发送相应的 PB 消息到车机，车机在收到该消息后，转而发送标准的 HFP AT Command 来接听电话。
- 电话成功接听后，车机更新状态到手机。
- 电话结束后，用户可以点击挂断按键，来通知车机转发 HFP 标准的结束通话命令。
- 最后同样更新电话状态到 CarLife 端。

9.3. 接口定义

9.3.1. 蓝牙电话主要功能

- 来电接听 (Answer An Incoming Call)
- 来电拒接 (Reject An Incoming Call)
- 去电呼叫 (Place A Call With Phone Number)
- 中止电话 (Terminate Call Process)
- 发送 DTMF 码 (Transmit DTMF Codes)
- 来电状态通知
- 去电状态通知
- 电话接通或者挂断通知
- 静音 MIC
- 车机在完成和 CarLife 的握手后，如果发现蓝牙已经连接好，车机将使能 CarLife Internal Telephone UI 功能，并阻止车机原生的蓝牙电话 APP 在系统前台，并将需要的电话事件通过 USB/WIFI 协议发送到手机 CarLife。
- 如果 Carlife 退出，车机需要能够恢复到使用原生的蓝牙电话 APP 来进行电话的有关操作。
- 三方通话功能不被支持，如果在通话的过程中，收到新的来电信息，需要使用车机原生的电话 APP 来处理
- 部分车机支持电话有关的 Hardkey，当相关的按键被按下时，对应的 keycode 需要发送到手机 CarLife。但是动作的执行还是车机自己完成。

9.3.2. 具体接口定义

9.3.2.1.MSG_CMD_BT_HFP_REQUEST

该消息由手机发送到车机，用来请求车机通过自身的蓝牙 HFP 协议，来完成电话有关的操作。具体消息中的字段定义如下表所示。

Field Name	Type	Optional/ Required	Description
command	Integer	Required	指明哪一个电话操作命令将被发送
phonenum	String	Optional	当需要呼出一个电话时，该字段需要指明呼叫的电话号码
dtmfcode	Integer	Optional	指明发送的 DTMF 编码

表 9.3-1

主要命令包括：

Name	Value	Usage
PLACE_OUTGOING_CALL	1	拨打一个电话
TERMINATE_CALL	2	挂断电话
ANSWER_CALL	3	接听电话

REJECT_CALL	4	拒接来电
SEND_DTMF	5	发送 DTMF 编码
MUTE_MIC	6	静音本地 MIC
UNMUTE_MIC	7	取消对本地 MIC 的静音

表 9.3-2

9.3.2.2.MSG_CMD_BT_HFP_INDICATION

该消息从车机发送到手机，用来告诉手机蓝牙连接后的电话状态。具体消息中的字段定义如下表所示。

Field Name	Type	Optional/ Required	Description
State	Integer	Required	指明电话的状态，具体电话状态可以参考表 11-4
Phonenum	String	Optional	当收到来电时，该字段指明来电的号码
Name	String	Optional	未使用字段
Address	String	Optional	当收到电话状态消息时，需要指明该电话状态来自如哪个蓝牙设备（地址）。车机需要填写该字段，告诉手机该状态变化来自哪个蓝牙地址。

表 9.3-3

主要的状态包括：

Name	Value	Usage
INCOMING_CALL	1	指明新的来电到来
OUTGOING_CALL	2	指明一个新的呼叫成功
CALL_ACTIVE	3	指明电话接通
CALL_INACTIVE	4	指明通话结束
MULTICALL_ACTIVE	5	指明有三方通话正在进行

表 9.3-4

9.3.2.3.MSG_CMD_BT_HFP_CONNECTION

当手机和车机的 CarLife USB 有线连接成功时候，当蓝牙 HFP 的连接状态发生改变时，该消息用来告诉手机蓝牙的连接状态。

具体消息中的字段定义如下表所示。

Field Name	Type	Optional/ Required	Description
State	Integer	Required	当蓝牙 HFP 连接状态发生变化时，该字段指示蓝牙 HFP 的具体连接状态，由表 11-6 定义
Address	String	Optional	被连接或者断开的目标设备的地址
Name	String	Optional	未使用字段

表 9.3-5

主要的状态定义包括：

Name	Value	Usage
HFP_DISCONNECTED	0	指明 HFP 的连接断开
HFP_CONNECTING	1	指明 HFP 正在连接
HFP_CONNECTED	2	指明 HFP 连接成功

表 9.3-6

9.3.2.4.MSG_CMD_BT_HFP_RESPONSE

该消息由车机发送给手机，用来反馈和告诉手机前面的请求执行的结果，手机端根据执行的结果来更新有关的界面状态。

具体消息中的字段定义如下表所示。

Field Name	Type	Optional/ Required	Description
status	Integer	Required	指明指定命令的执行结果。 0: 命令执行失败 1: 命令执行成功
cmd	Integer	Required	指明是执行的哪一个命令，具体命令参见下面的表格
dtmfcode	Integer	Optional	如果是发送 DTMF code 成功，需要反馈是发送的哪一个 DTMFcode,包括数字 0-9, *,#

表 9.3-7

主要命令包括：

Name	Value	Usage
PLACE_OUTGOING_CALL	1	拨打一个电话
TERMINATE_CALL	2	挂断电话
ANSWER_CALL	3	接听电话
REJECT_CALL	4	拒接来电
SEND_DTMF	5	发送 DTMF 编码
MUTE_MIC	6	静音本地 MIC
UNMUTE_MIC	7	取消静音

表 9.3-8

9.3.2.5.MSG_CMD_BT_HFP_STATUS_REQUEST

该命令从手机发送到车机，用来获取指定的状态，目前主要用来获取车机端 MIC 是否 mute 得状态。

Field Name	Type	Optional/ Required	Description
type	Integer	Required	指定获取哪一种状态。 1: 获取 MIC 的静音状态

表 9.3-9

9.3.2.6.MSG_CMD_BT_HFP_STATUS_RESPONSE

该命令从车机反馈给手机，当车机端收到手机的 STATUS_REQUEST 后，需要反馈指定的状态给手机，目前主要用来反馈车机 MIC 是否 Mute 的状态。

Field Name	Type	Optional/ Required	Description
status	Integer	Required	指明指定命令的执行结果。 0: MIC 状态是 Unmute 1: MIC 状态是 Mute
type	Integer	Required	指明反馈的是哪一种状态。 1:反馈的是 MIC 的状态

9.3.2.7.MSG_CMD_BT_IDENTIFY_RESULT_IND

该命令将从手机发送到车机，用来告诉车机，与车机蓝牙的 HFP 连接，已经成功建立或者失败建立。只有当连接建立或者断开的时候，该消息才会发送。当车机收到 state 为 SUCCESS 时，车机需要屏蔽原生的蓝牙电话功能，CarLife 内置蓝牙电话将接管电话的操作。当车机收到 state 为 FAILURE 时，需要启用原生的蓝牙电话来控制电话功能。

具体格式见下表：

Field Name	Type	Optional/ Required	Description
state	Integer	Required	该字段包括两种状态：SUCCESS, FAILURE. SUCCESS = 1: 当手机和车机的蓝牙 HFP 连接上的时候，该状态将会被发送给车机，告诉车机，蓝牙已经连接好了。 FAILURE = 0: 当手机和车机的蓝牙 HFP 断开，该状态将会被发送给车机，通知车机，与车机的 HFP 蓝牙连接断开了
address	String	Required	被连接或者断开的目标设备的地址，一般是车机的蓝牙地址

9.3.2.8.MSG_CMD_BT_START_IDENTIFY_REQ

该消息从车机发送到手机，用来告诉手机，开始标定是否连接上的蓝牙设备是车机。标定的结果，通过 MSG_CMD_BT_IDENTIFY_RESULT_IND 给出。具体消息格式如下：

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

address	String	Required	需要标定的车机的蓝牙地址。
---------	--------	----------	---------------

9.4. 重要场景交互时序

9.4.1. 去电呼叫场景

一个典型的去电呼叫场景的时序如下，该流程适用于用户通过 CarLife 的 Internal Telephone UI 来拨打一个电话。详细流程可以参考时序里面的描述。

此场景只适用于 Iphone 手机的 Internal Telephone UI,如果车机连接的是 Android 手机，去电时**大部分手机端**会直接进行拨号,手机端不会通过 MSG_CMD_BT_HFP_REQUEST 消息通知车机端,此时车机端**可以**通过蓝牙获取到通话状态,并通过 MSG_CMD_BT_HFP_INDICATION 消息通知到手机端。

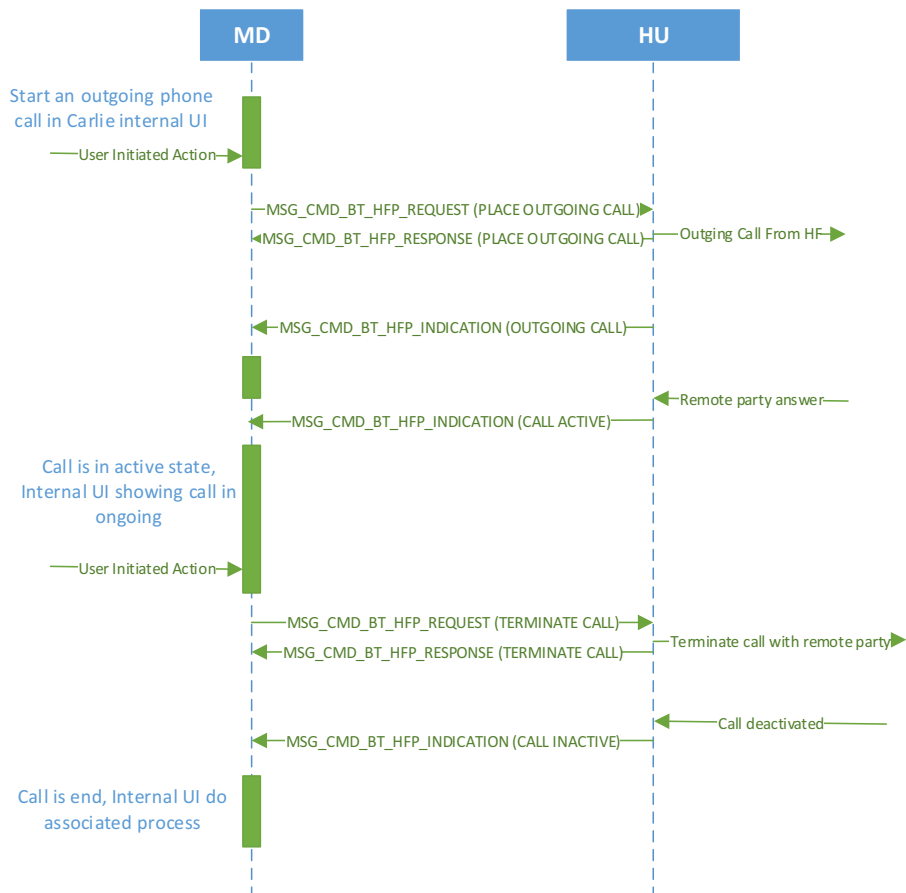


图 9.4-1

9.4.2. 通话过程中发 DTMF 编码

通话过程发送 DTMF 的时序图如下所示，该流程适用于，用户在使用 CarLife Internal Telephone UI 来通电话的过程中，发送 DTMF 编码。详细流程见下图所示。

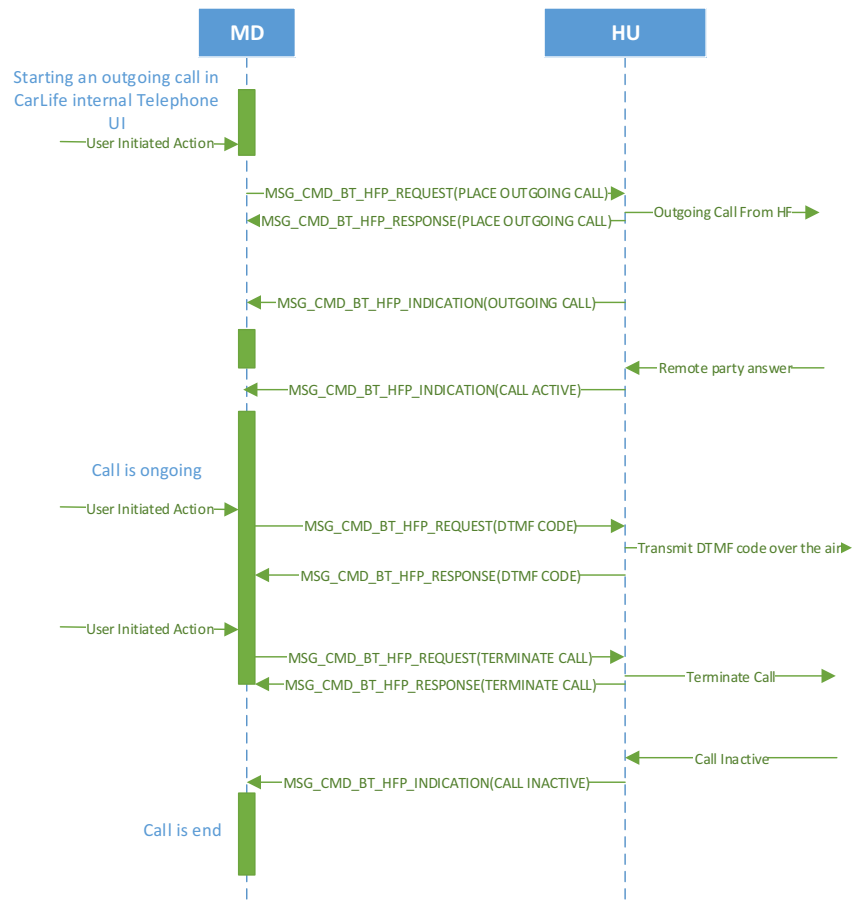


图 9.4-2

9.4.3. 蓝牙状态改变场景

只要在 CarLife USB 连接成功，并且协议版本匹配之后，蓝牙 HFP 的连接状态都需要从车机端发送到手机端，手机 CarLife 会以此来确定是否 CarLife Internal Telephone UI 能否启用。当 HFP 连接断开后，蓝牙电话有关的功能会无法生效。

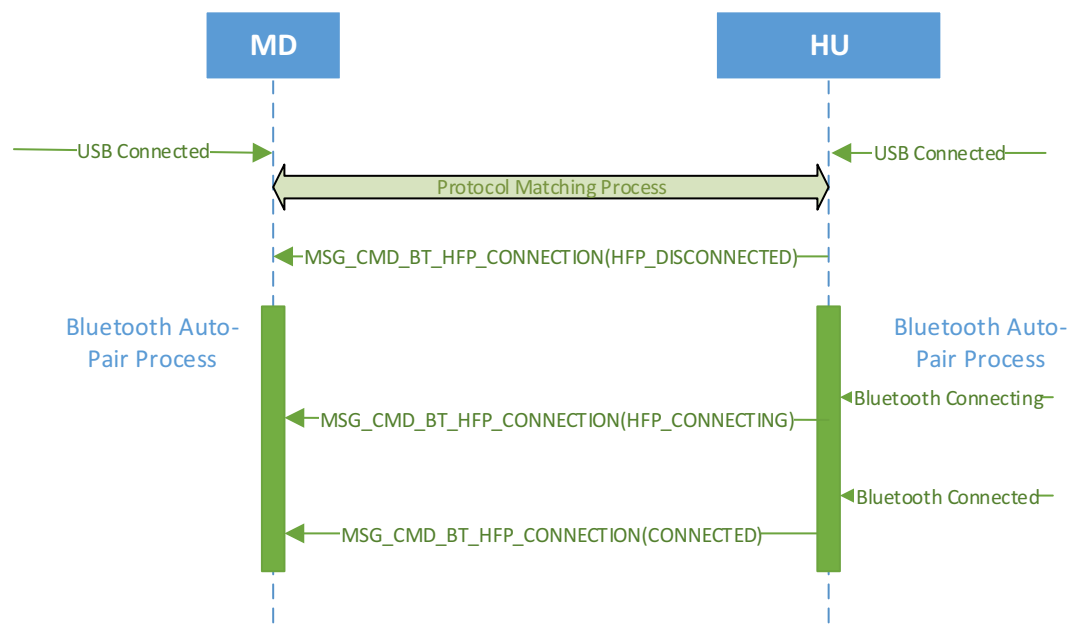


图 9.4-3

10. 通信协议

10.1. service type 命名规则

所有通道的消息结构中都有一个 **service type** 字段,用来唯一标识该条消息,对该字段的命令也有一定的要求。**service type** 字段宏定义命名规则为: **MSG_**所在通道名称_**service type** 的具体作用关键词。例如 **Android** 移动设备发送给车机的 **Android** 系统信息对应的宏定义名称为: **MSG_CMD_MD_INFO**。

service type 前面两个字节用来表示不同的通道或者主要服务的 ID, 后面两个字节用来表示具体的 **service type** 值, 后面两个字节的最前面 1 位为 0 表示该条消息是移动设备发送车机的, 为 1 表示该条消息是车机发送给移动设备的。前两个字节 ID 如表所示。

表 10.1-1 CarLife 各通道 service type 前两个字节 ID

通道名称	前两个字节
Cmd	0x0001
Video	0x0002
Music	0x0003
TTS	0x0004

VR	0x0005
Touch	0x0006

10.2. 控制消息

CarLife 的控制消息通过 Cmd 通道来传输，主要是传输各个模块之间的同步消息、简单数据交互等。

10.2.1. 包结构

CarLife 控制消息结构如下图所示：

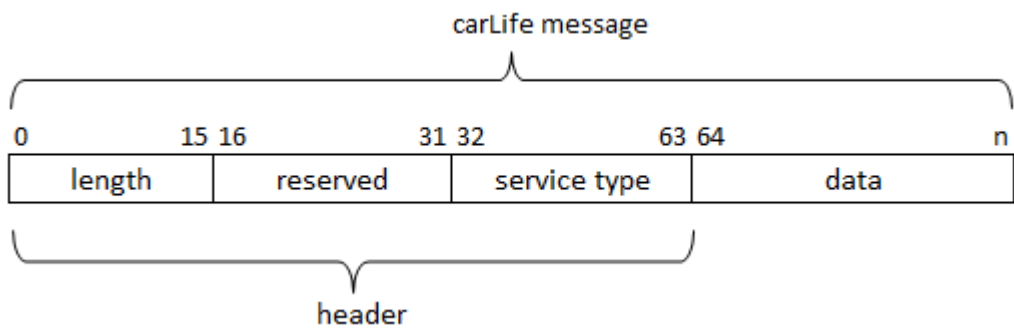


图 10.2-1: CarLife 控制消息结构

CarLife 控制消息 header 的长度固定，为 8 个字节，各字段定义如表 10.2-1 所示：

表 10.2-1: CarLife 控制消息各字段定义

字段名称	位偏移	大小(bit)	说明
length	0~15	16	data 字段用 PB 序列化后的长度
reserved	16~31	16	保留字段
service type	32~63	32	消息服务类型，详见相关消息体
data	64~n	length*8	service type 字段的参数，用 PB 序列化后的字节流

10.2.2. 相关消息体

控制消息的所有 service type 对应的 data 字段均用 PB 序列化。车机端和移动设

备端 service type 及对应的 data 字段定义如表 10.2-1 所示。

表 10.2-1: CarLife 控制消息的 service type 及 data 字段定义

字段宏定义名称	service type	data 字段 pb 结构	流向	含义及作用	应用场景
MSG_CMD_HU_PROTOCOL_VERSION	0x00018001	<pre> message CarlifeProtocolVersion { required int32 majorVersion = 1; required int32 minorVersion = 2; } enum PBVersion { [majorVersion:1, minorVersion:0], [majorVersion:2, minorVersion:0] } </pre>	HU->MD	连接成功以后，第一交互的命令必须是这个命令，车机将自身的协议版本信息发给移动设备。移动设备根据版本号判断是否兼容。版本信息采用主次版本号形式。 majorVersion: 为主版本号； minorVersion: 为次版本号。 所有 PBVersion 版本见 enum 定义，当前最新版本为 [majorVersion:2, minorVersion:0]。	连接建立成功以后车机端发送给移动设备的第一条消息。
MSG_CMD_PROTOCOL_VERSION_MATCH_STATUS	0x00010002	<pre> message CarlifeProtocolVersionMatchStatus { required int32 matchStatus = 1; } </pre>	MD->HU	移动设备根据车机的版本信息判断版本是否兼容。匹配状态如下： VERSION_MATCH = 1; VERSION_NOT_MATCH = 2;	在接受到 MSG_CMD_HU_PROTOCOL_VERSION 消息以后，移动设备发送给车机的第一条消息。
MSG_CMD_HU_INFO	0x00018003	<pre> 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; } </pre>	HU->MD	车机发送给移动设备的车机系统信息，其中 os 字段表示车机操作系统的类型，可以为：QNX、Linux、Wince、Android。	连接建立成功以后车机主动将相关信息发送给移动设备端。详见统计信息声明章节。
MSG_CMD_MD_INFO	0x00010004	<pre> 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; } </pre>	MD->HU	移动设备发送给车机的移动设备系统信息，其中 os 字段表示移动设备操作系统的类型，可以为：Android、IOS。其中，release 字段	连接建立成功以后移动设备主动将相关信息发送给车机端。

		<pre>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 = 24; }</pre>		描述的是 CarLife 手机端的版本号，value 值是以“CarLifePhoneVersion:”为前缀的字符串。	
MSG_CMD_HU_BT_OOB_INFO	0x00018005	<pre>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; }</pre>	HU->MD	车机发送给移动设备的蓝牙自动配对的信息。	CarLife 握手成功后，车机通过发送这条消息来开始蓝牙自动配对过程，具体场景请参考 蓝牙自动配对 章节。
MSG_CMD_MD_BT_OOB_INFO	0x00010006	<pre>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; }</pre>	MD->HU	移动设备发送给车机的蓝牙自动配对的信息。	手机在收到车机发送过来的配对信息之后需要通过这条消息来回复他的蓝牙信息，具体场景请参考 蓝牙自动配对 章节。
MSG_CMD_VIDEO_ENCODER_INIT	0x00018007	<pre>message CarlifeVideoEncoderInfo { required int32 width = 1; required int32 height = 2; required int32 frameRate = 3; }</pre>	HU->MD	车机发送给移动设备的视频截取初始化请求。	详见 视频消息 章节。
MSG_CMD_VIDEO_	0x0001	<pre>message CarLifeVideoEncoderInfo {</pre>	MD->	移动设备发送给车机的视频截取初始	详见 视频消息 章

ENCODER_INIT_DONE	0008	required int32 width = 1; required int32 height = 2; required int32 frameRate = 3; }	HU	化成功消息，返回视频的分辨率。	节。
MSG_CMD_VIDEO_ENCODER_START	0x00018009	无	HU->MD	车机发送给移动设备开始传输视频的消息。	详见 视频消息 章节。
MSG_CMD_VIDEO_ENCODER_PAUSE	0x0001800A	无	HU->MD	车机发送给移动设备暂停视频传输的消息。移动设备端接收到该消息后，处理方式与切换到后台时处理方式相同。	详见 视频消息 章节。
MSG_CMD_VIDEO_ENCODER_RESET	0x0001800B	无	HU->MD	车机发送给移动设备重新初始化视频传输的消息。	详见 视频消息 章节。
MSG_CMD_VIDEO_ENCODER_FRAME_RATE_CHANGE	0x0001800C	message CarlifeVideoFrameRate { required int32 frameRate = 1; }	HU->MD	车机发送给移动设备的视频帧率变更请求。	详见 视频消息 章节。
MSG_CMD_VIDEO_ENCODER_FRAME_RATE_CHANGE_DONE	0x0001000D	message CarlifeVideoFrameRate { required int32 frameRate = 1; }	MD->HU	移动设备发送给车机的视频帧率变更成功消息，返回变更后的帧率。	详见 视频消息 章节。
MSG_CMD_PAUSE_MEDIA	0x0001800E	无	HU->MD	车机发送给移动设备的暂停播放Media的消息。	已废弃
MSG_CMD_CAR_VELOCITY	0x0001800F	message CarlifeCarSpeed { required uint32 speed = 1; optional uint64 timeStamp = 2; }	HU->MD	车机发送给移动设备的当前速度信息。	详见 车载数据使用及建议 章节。
MSG_CMD_CAR_GPS	0x00018010	message CarLifeCarGps { 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	HU->MD	车机发送给移动设备的当前位置信息。 支持 WGS84 和 GCJ02 坐标系。 默认发送频率为 1HZ	详见 车载数据使用及建议 章节。

		<pre> 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>			
MSG_CM D_CAR_GY ROSCOPE	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-> MD	车机发送给移动设备的当前汽车陀螺仪信息。 当车机采用单轴陀螺仪时，将 gyroX 设置为单轴陀螺仪的值，gyroY 和 gyroZ 赋值为 0。 陀螺仪的值为角速度，单位为度/秒。 gyroX: 车机 Y 轴水平面上的投影与正北方向的夹角。 默认发送频率为 10HZ	详见 车载数据使用及建议 章节。
MSG_CM D_CAR_AC CELERATI ON	0x0001 8012	<pre> message CarlifeAcceleration { required doubleaccX = 1; required doubleaccY = 2; required doubleaccZ = 3; optional uint64 timeStamp = 4; } </pre>	HU-> MD	车机发送给移动设备的当前汽车加速度信息。 加速度计值的单位是米/秒的平方 accX: 水平方向加速度，右为正。 accY: 水平垂直方向，前为正。 accZ: 空间垂直方向，天空方向为正。 默认发送频率为 10HZ	详见 车载数据使用及建议 章节。
MSG_CM D_CAR_OI L	0x0001 8013	<pre> message CarlifeOil { required int level = 1; required int range = 2; required bool lowFuleWarning = 3; } </pre>	HU-> MD	车机发送给移动设备的当前汽车油耗信息。	请参考 车载数据使用及建议 章节。
MSG_CM D_SCREEN _ON	0x0001 0018	无	MD-> HU	移动设备通知车机移动设备屏幕被点亮。	暂时没有使用

MSG_CM D_SCREEN _OFF	0x0001 0019	无	MD-> HU	移动设备通知车机 移动设备屏幕关 闭。	暂时没有 使用
MSG_CM D_SCREEN _USERPRE SENT	0x0001 001A	无	MD-> HU	移动设备通知车机 移动设备解锁。	暂时没有 使用
MSG_CM D_FOREG ROUND	0x0001 001B	无	MD-> HU	移动设备通知车机 移动设备被切换到 前台。	请参考手 机端锁屏 或者前后 台切换
MSG_CM D_BACKG ROUND	0x0001 001C	无	MD-> HU	移动设备通知车机 移动设备被切换到 后台。	请参考手 机端锁屏 或者前后 台切换
MSG_CM D_LAUNC H_MODE_ NORMAL	0x0001 801D	无	HU-> MD	车机启动移动设备 CarLife 进入 Home 界面。	无
MSG_CM D_LAUNC H_MODE_ PHONE	0x0001 801E	无	HU-> MD	车机启动移动设备 CarLife 进入 Phone 界面。	无
MSG_CM D_LAUNC H_MODE_ MAP	0x0001 801F	无	HU-> MD	车机启动移动设备 CarLife 进入 Map 界 面。	无
MSG_CM D_LAUNC H_MODE_ MUSIC	0x0001 8020	无	HU-> MD	车机启动移动设备 CarLife 进入 Music 界面。	无
MSG_CM D_GO_TO _DESKTOP	0x0001 0021	无	MD-> HU	点击移动设备端 CarLife 退出到车载 系统按钮时，发送 该条命令。	无
MSG_CM D_MIC_RE CORD_WA KEUP_STA RT	0x0001 0022	无	MD-> HU	移动设备通知车机 进入唤醒状态，开 始录音（以 16k 采 样率）。	语音唤醒 启动时， 由手机端 发出
MSG_CM D_MIC_RE CORD_EN D	0x0001 0023	无	MD-> HU	移动设备通知车机 暂停录音。	语音唤醒 或者语音 识别结束 时，由手 机端发出

MSG_CMD_MIC_RECORD_START	0x00010024	无	MD->HU	移动设备通知车机进入识别状态，开始录音（以 16k 采样率）。	语音识别启动时，由手机端发出
MSG_CMD_GO_TO_FOREGROUND	0x00018025	无	HU->MD	点击启动 CarLife 按钮时，车机端发送该条消息给移动设备端将 CarLife 切回到前台。	目前只有 AOA 连接才会用到该消息，ADB 连接使用 <u>ADB 命令</u> ， <u>iOS 有线连接</u> 需要发送相关指令到底层，详见相关章节。
MSG_CMD_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_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; PHONE_STATUS_CONNECTED=3; } enum NaviState { NAVI_STATUS_IDLE =0; NAVI_STATUS_START=1;</pre>	MD->HU	该消息主要用于移动设备与车机互联时同步状态：移动设备通知车机音乐，语音，导航等模块状态改变。 备注： 1. <pre>enum ModuleId { CARLIFE_EDOG_MODULE_ID=8; CARLIFE_CRUISE_ID=9; }</pre> 这两个 MODULE_ID 为 (PB_VERSION 2.0 新增)	当移动设备端相关模块的状态改变时发送给车机端，各模块状态见 <u>enum</u> 类型中的描述。

		<pre> } 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_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_VEHICLE_MIC =0; MIC_STATUS_USE_MOBILE_MIC =1; } enum MediaPCMState { MEDIAPCM_STATUS_IDLE=0; MEDIAPCM_STATUS_RUNNING=1; } enum EDogState { EDOG_STATUS_IDLE =0; EDOG_STATUS_RUNNING =1; } enum CruiseState { </pre>			
--	--	---	--	--	--

		<pre> CRUISE_STATUS_IDLE = 0; CRUISE_STATUS_RUNNING = 1; } message CarlifeModuleStatusList { required int32 cnt = 1; repeated com.baidu.carlife.protobuf. CarlifeModuleStatus moduleStatus = 2; } message CarlifeModuleStatus { required int32 moduleID = 1; required int32 statusID = 2; } </pre>			
MSG_CMD_STATISTIC_INFO	0x00018027	<pre> message CarlifeStatisticsInfo { required string cuid = 1; required string versionName = 2; required int32 versionCode = 3; required string channel = 4; required int32 connectCount = 5; required int32 connectSuccessCo unt = 6; required int32 connectTime = 7; optional string crashLog = 8; } </pre>	HU->MD	统计信息: cuid: HU 的唯一标识码 versionName: HU 的版本名称 versionCode: 车机端 CarLife 的版本号 channel: 车厂渠道号, 由百度分配 connectCount: 连接次数, 由车机端 CarLife 动态保存, 连接成功后, 该参数需要清零 connectSuccessCount: 连接成功次数, 一般设置为 1 connectTime: 连接过程花费时长 crashLog: 车机端上次 crash 关键日志	该条消息在连接成功以后必须有车机端发送给移动设备端, 而且必须保证 cuid 在每一台车机中都不一样, 否则会引起移动设备端自动断开, 请务必按照要求填写各个字段, 并发送给移动设备端。详见统计信息声明章节。
MSG_CMD_MODULE_CONTROL	0x00018028	<pre> enum ModuleId { CARLIFE_PHONE_MODULE_ID=1; } </pre>	HU->MD	该消息主要用于车机端对移动设备端各模块进行控制。车机端可以通过该消息控制移动设备	当车机端需要控制各模块相关功能时, 设置

	<pre> 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_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_RUNNING =1; } enum ConnectState { CONNECT_STATUS_ADB=1, CONNECT_STATUS_AOA=2, CONNECT_STATUS_NCM_ANDROID=3, CONNECT_STATUS_NCM_IOS =4, </pre>	端音乐，语音，导航等模块。 备注： 1. CARLIFE_MEDIAPCM_MODULE_ID 目前仅用于 iOS 设备。某些支持 iPod 音乐播放的车机，当车机端 CarLife 切后台 iPod 音乐播放在前台时，可以使用该命令暂停 MD 向 HU 传输音乐 PCM 流，避免底层带宽不够用。 2. enum ModuleId { CARLIFE_EDOG_MODULE_ID=8; CARLIFE_CRUISE_ID=9; } 这两个 MODULE_ID 为 (PB_VERSION 2.0 新增)	相关状态值，并将该消息发送给移动设备端。
--	--	---	-----------------------------

		<pre> CONNECT_STATUS_WIFI=5, } enum MicState { MIC_STATUS_USE_VEHICLE_MIC =0; MIC_STATUS_USE_MOBILE_MIC =1; } 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 CarlifeModuleStatus { required int32 moduleID = 1; required int32 statusID = 2; } </pre>			
MSG_CM D_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-> MD	通过档位信息标示倒车信号。 目前车机只发送 REVERSE 档位; 非倒车时发送 DRIVE 档位。 (该消息后期可用于用户驾驶行为习惯分析)	请参考 车载数据使用和建议 章节。

MSG_CM D_NAVI_N EXT_TURN _INFO	0x0001 0030	<pre> message CarlifeNaviNextTurnInfo { required ActionType action = 1; required TurnKind nextTurn = 2; required string roadName = 3; required int32 totalDistance = 4; required int32 remainDistance = 5; required byte turnIconData = 6; } </pre>	MD-> HU	移动设备发送给车机的 Turn-By-Turn 导航信息。该消息用于车机在主屏之外的其他的屏幕显示相关的诱导信息	该消息具体定义和相关枚举类型定义请参考 导航 HUD 信息章节
MSG_CM D_CAR_DATA_SUBSCRIBE	0x0001 0031		MD-> HU	移动设备发送给车机的订阅车身数据的消息，消息内容为空。（查看该车机支持的数据类型，以及各类型数据产生的频率）	请参考 车载数据使用和建议 章节。
MSG_CM D_CAR_DATA_SUBSCRIBE_DONE	0x0001 8032	<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.protobuf.CarLifeVehicleInfo vehicleInfo = 2; } message CarlifeVehicleInfo { required int32 moduleID = 1; required int32 supportFlag = 2; required uint32 frequency = 3; } </pre>	HU-> MD	车机针对移动设备订阅车身数据的反馈信息。 supportFlag: 该车机是否支持当前车身数据。（1 为支持，0 为不支持；对于 GPS_Module，1 为 WGS-84 坐标系，2 为 GCJ-02 坐标系） Frequency: 车机产生该类型数据的频率。（当不支持时设置频率为 0）	请参考 车载数据使用和建议 章节。
MSG_CM D_CAR_DATA_SUBSCRIBE_START	0x0001 0033	<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 </pre>	MD-> HU	移动设备要求车机开始发送订阅的车身信息，如移动设备需要订阅多个类型数据时，则会多次发送该消息。消息内只包含一个 CarLifeVehicleInfo 。	请参考 车载数据使用和建议 章节。

		<pre> { required int32 moduleID = 1; optional int32 supportFlag = 2; required uint32 frequency= 3; } </pre>			
MSG_CMD_CAR_DATA_SUBSCRIBE_STOP	0x00010034	<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; optional uint32 frequency= 3; } </pre>	MD->HU	移动设备要求车机停止发送订阅的车身信息，如移动设备需要取消订阅多个类型数据时，则会多次发送该消息。消息内只包含一个CarLifeVehicleInfo。	请参考 车载数据使用和建议 章节。
MSG_CMD_MEDIA_INFO	0x00010035	<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 playlistNum = 7; required string songId = 8; required PlayMode mode = 9; } </pre>	MD->HU	移动设备发送给车机的关于歌曲信息的内容	当音乐播放状态有变更时，手机端会发送该消息，车机端可以根据自己的业务需求接收该消息作相关的展示，请参考当前歌曲播放信息章节
MSG_CMD_MEDIA_PROGRESS_BAR	0x00010036	<pre> message CarlifeMediaProgressBar { required int32 progressBar = 1; } </pre>	MD->HU	移动设备端发送给车机端的音乐播放的进度条信息（单位 ms）	音乐播放时手机端会每秒发一次
MSG_CMD_CONNECT_EXCEPTION	0x00010037	<pre> message CarlifeConnectException { required int32 exceptionType = 1; } TYPE_ENCODER_ERROR = 1; </pre>	MD->HU	移动设备端发送给车机端连接异常的消息（比如视频编码器初始化失败，Android5.0 以上用户拒绝屏幕共享	TYPE_ENCODER_ERROR 表示编码器异常； TYPE_PER

		TYPE_PERMISSION_DENIED = 2; TYPE_SCREENSHARE_REQUEST = 3;		等)	MISSION_DENIED 表示申请屏幕共享权限的时候被用户拒绝了; TYPE_SCREENSHARE_REQUEST 表示 Android5.0 以上移动设备正在请求屏幕共享, 移动设备上会有一个请求弹窗, 车机收到该消息可以给出一个相应提示, 让用户响应移动设备弹窗请求。
MSG_CMD_REQUEST_GO_TO_FOREGROUND	0x00010038	无	MD->HU	移动设备端发送给车机端要求车机重新把移动设备端 CarLife 应用拉起的消息。	有些情况下需要车机端执行命令将 CarLife 应用重新拉回到前台, 此时移动设备端将会发送该条消息给车机端。
MSG_CMD_UI_ACTION_SOUND	0x00010039	无	MD->HU	移动设备通知车机播放屏幕按键音	无
MSG_CMD_BT_HFP_REQUEST	0x00010040	Enum RequestType { REQUEST_STARTCALL=1, REQUEST_TERMINATECALL=2, REQUEST_ANSWERCALL=3, REQUEST_REJECTCALL=4, REQUEST_DTMFCODE=5, REQUEST_MUTEMIC=6 }	MD->HU	移动设备发送给车机, 用来控制车机的蓝牙电话功能	在车机使用 CarLife Internal Telephone UI 的时候, 将被用来发送电话的请求指令。

		<pre>message CarlifeBTHfpRequest { required RequestType command = 1; optional string phonenum = 2; optional int32 dtmfcode = 3; }</pre>			具体场景请参考 蓝牙电话 章节。
MSG_CM D_BT_HFP _INDICATI ON	0x0001 8041	<pre>Enum IndicationType { INDICATION_NEWCALL=1, INDICATION_OUTCALL=2, INDICATION_CALLACTIVE=3, INDICATION_NOCALLACTIVE=4 INDICATION_MULTICALLACTIVE = 5, INDICATION_MULTICALLINACTIV E = 6 } message CarlifeBTHfpIndication { required IndicationType state = 1; optional string phonenum = 2; optional string name = 3; optional string address = 4; }</pre>	HU-> MD	车机发送给移动设备，用来告诉移动设备蓝牙电话的状态信息。其中需要指明该状态是来自那个蓝牙设备（地址）。主要考虑到有的车机能够支持同时连接多个蓝牙设备。	在车机使用 CarLife Internal Telephone UI 的时候，将被用来发送电话的状态指令。具体场景请参考 蓝牙电话 章节。
MSG_CM D_BT_HFP _CONNEC TION	0x0001 8042	<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-> MD	车机发送该消息到移动设备用来告诉移动设备关于蓝牙的连接状态(HFP 规范)	在车机使用 CarLife Internal Telephone UI 的时候，将被用来更新蓝牙 HFP 的连接状态信息。CarLife 连接成功后，车机都需要发送这条消息来告诉手机蓝牙 HFP 的连接状态。具体场景请参考 蓝牙电话 章节。
MSG_CM D_CARLIF E_DATA_S UBSCRIBE	0x0001 8043	<pre>enum ModuleID { CARLIFE_DATA_TURNBYTURN=0 ; CARLFIE_DATA_ASSISTANTGUID E=1; } message</pre>	HU-> MD	车机发送给移动设备的订阅移动设备 CarLife 相关数据的信息。（查看该手机 CarLife 版本是否支持该指定类型的数据提供）	车机端需要使用 CarLife 里的导航模块提供 HUD 信息或者电子眼信息

		<pre> CarlifeSubscribeMobileCarLifeInfoList { required int32 cnt = 1; repeated com.baidu.carlife.protobuf.CarlifeSub scribeMobileCarLifeInfo = 2; } message CarlifeSubscribeMobileCarLifeInfo { required int32 moduleID = 1; required boolsupportFlag = 2; } </pre>		CARLIFE_DATA_TURNBYTURN 是 简 易 诱导消息 CARLFIE_DATA_ASSISTANTGUIDE 是 辅 助诱导消息	时，可通 过该消息 发 起 订 阅，手机 端 会 回 复 MSG_CM D_CARLIF E_DATA_S UBSCRIBE _DONE
MSG_CM D_CARLIF E_DATA_S UBSCRIBE _DONE	0x0001 0044	<pre> enum ModuleID { CARLIFE_DATA_TURNBYTURN=0 ; CARLFIE_DATA_ASSISTANTGUID E=1; } message CarlifeSubscribeMobileCarLifeInfoList { required int32 cnt = 1; repeated com.baidu.carlife.protobuf.CarlifeSub scribeMobileCarLifeInfo = 2; } message CarlifeSubscribeMobileCarLifeInfo { required int32 moduleID = 1; required boolsupportFlag = 2; } </pre>	MD-> HU	移动设备针对车机 订阅 CarLife 相关数 据的反馈信息。 CARLIFE_DATA_TURNBYTURN 是 简 易 诱导消息 CARLFIE_DATA_ASSISTANTGUIDE 是 辅 助诱导消息 supportFlag: 该移动 设备的 CarLife 版本 是否支持当前数据 的提供。	对车机端 发 的 MSG_CM D_CARLIF E_DATA_S UBSCRIBE 消息的应 答
MSG_CM D_CARLIF E_DATA_S UBSCRIBE _START	0x0001 8045	<pre> enum ModuleID { CARLIFE_DATA_TURNBYTURN=0 ; CARLFIE_DATA_ASSISTANTGUID E=1; } message CarlifeSubscribeMobileCarLifeInfoList { required int32 cnt = 1; repeated com.baidu.carlife.protobuf.CarlifeSub scribeMobileCarLifeInfo = 2; } </pre>	HU-> MD	车机端要求手机端 开始发送订阅的相 关消息。 CARLIFE_DATA_TURNBYTURN 是 简 易 诱导消息 CARLFIE_DATA_ASSISTANTGUIDE 是 辅 助诱导消息	

		<pre> message CarlifeSubscribeMobileCarLifeInfo { required int32 moduleID = 1; required boolsupportFlag = 2; } </pre>			
MSG_CM D_CARLIF E_DATA_S UBSCRIBE _STOP	0x0001 8046	<pre> enum ModuleID { CARLIFE_DATA_TURNBYTURN=0 ; CARLFIE_DATA_ASSISTANTGUID E=1; } message CarlifeSubscribeMobileCarLifeInfoList { required int32 cnt = 1; repeated com.baidu.carlife.protobuf.CarlifeSub scribeMobileCarLifeInfo = 2; } message CarlifeSubscribeMobileCarLifeInfo { required int32 moduleID = 1; required boolsupportFlag = 2; } </pre>	HU-> MD	车机端要求手机端停止发送订阅的相关消息。 CARLIFE_DATA_TURNBYTURN 是 简 易 诱导消息 CARLFIE_DATA_ASSISTANTGUIDE 是 辅 助诱导消息	
MSG_CM D_NAVI_A SSITANTG UIDE_INF O	0x0001 0047	<pre> message CarLifeNaviAssitantGuideInfo { required ActionType action = 1; required AssistantType assistantType =2; required TrafficSignType trafficSignType =3; required int32 totalDistance = 4; required int32 remainDistance = 5; required int32 cameraSpeed = 6; } </pre>	MD-> HU	移动设备发送给车机的导航辅助诱导信息。该消息用于车机在主频之外的其他的屏幕显示相关的辅助诱导信息	该消息结构具体定义和相关枚举的定义请参考 导航 HUD 信息 章节
MSG_CM D_HU_AU THEN_RE QUEST	0x0001 8048	<pre> message CarlifeAuthenRequest { required string randomValue = 1; } </pre>	HU-> MD	车机 CarLife 在本地随机产生一个随机字符串，并将该值发送给手机端	车 机 端 CarLife 对 手 机 端 CarLife 进 行 安全认

				CarLife;	证。
MSG_CMD_MD_AUTHEN_RESPONSE	0x00010049	message CarlifeAuthenResponse { required string encryptValue = 1; }	MD->HU	手机 CarLife 将接收到的车机随机数使用私有密钥进行加密;	车机端 CarLife 对手机端 CarLife 进行安全认证。
MSG_CMD_HU_AUTHEN_RESULT	0x0001804A	message CarlifeAuthenResult { required bool result = 1; }	HU->MD	车机端 CarLife 将验证结果发送给手机端 CarLife	车机端 CarLife 对手机端 CarLife 进行安全认证。
MSG_CMD_MD_AUTHEN_RESULT	0x0001004B	message CarlifeAuthenResult { required bool result = 1; }	MD->HU	手机端 CarLife 将验证结果发送给车机端 CarLife	手机端 CarLife 对车机端 CarLife 进行安全认证。
MSG_CMD_GO_TO_FOREGROUND_RESPONSE	0x0001004C	无	MD->HU	车机端发送 MSG_CMD_GO_TO_FOREGROUND 消息给移动设备端将 CarLife 切回到前台以后, 移动设备端如果在 5 秒内没有切换成功则发送该条消息。	无
MSG_CMD_START_BT_AUTOPAIR_REQUEST	0x0001004D	message CarlifeBTStartPairReq { required int32 ostype = 1; optional string address = 2; }	MD->HU	手机端发送给车机, 断开已经存在的 HFP 连接, 并开始蓝牙自动匹配的流程。如果是 IOS 则不需要开始自动匹配流程	具体可以参考章节 6.2.3 初始化蓝牙自动匹配
MSG_CMD_BT_HFP_RESPONSE	0x0001804E	Enum RequestType { REQUEST_STARTCALL=1, REQUEST_TERMINATECALL=2, REQUEST_ANSWERCALL=3, REQUEST_REJECTCALL=4, REQUEST_DTMFCODE=5, REQUEST_MUTEMIC=6 } message CarlifeBTHfpResponse { Required int32 status = 1; Required RequestTypecmd = 2; Optional int32 dtmfCode = 3; }	HU->MD	车机端在处理完成 HFP 的请求后, 需要通过该消息告诉手机该命令的执行结果。	

MSG_CMD_BT_HFP_STATUS_REQUEST	0x0001004F	Enum StatusType { STATUS_MIC=1 } message CarlifeBTHfpStatusRequest { Required StatusType type = 1; }	MD->HU	手机发送该消息来获取指定的状态，目前主要用来获取车机MIC的状态	
MSG_CMD_BT_HFP_STATUS_RESPONSE	0x00018050	Enum StatusType { STATUS_MIC=1 } message CarlifeBTHfpStatusResponse { Required int32status = 1; Required StatusType type = 2; }	HU->MD	当车机收到手机请求状态时,需要通过该消息将指定的状态反馈给手机端。目前主要用来反馈车机MIC的状态。	;
MSG_CMD_MD_FEATURE_CONFIG_REQUEST	0x00010051	无	MD->HU	双方握手认证通过后，手机端会发送该消息来请求车机端的功能定制项	
MSG_CMD_HU_FEATURE_CONFIG_RESPONSE	0x00018052	详细参数信息请参考列表 <u>10.2-2</u>	HU->MD	车机端根据自身情况进行功能定制，当前支持9种功能定制，分别是麦克风（VOICE_MIC）、语音唤醒（VOICE_WAKEUP）、蓝牙自动连接（BLUETOOTH_AUTO_PAIR）、蓝牙电话（BLUETOOTH_INTERNAL_UI）、焦点态（FOCUS_UI）、媒体采样率（MEDIA_SAMPLE_RATE）等，后续还可支持更多功能定制。参数的结构是键值对列表（List<Key-Value>）的PB结构，列表的子项不固定，HU根据需要的定制项传入相应的键值对即可。	
MSG_CMD_BT_START_IDENTIFY_REQUEST	0x00018053	message CarlifeBTStartIdentifyReq { Required Stringaddress = 1; }	HU->MD	车机发送该消息来让手机开始蓝牙标识过程	具体过程可以参考蓝牙自动匹配 6.2.3 章节的初始化流程
MSG	0x0001	message CarlifeBTIdentifyResultInd	MD->	手机发送给车机，	具体参考

_CMD_BT_IDENTIFY_RESULT_IND	0054	{ Required intstatus = 1; Required String address = 2; }	HU	告诉车机蓝牙标识的结果，同时当标识状态发生变化时，通知车机	章节 6.2.3 蓝牙自动匹配初始化流程
MSG_CMD_ERROR_CODE	0x00018055	message CarlifeErrorCode { required string errorCode = 1; }	HU->MD	车机发送该消息，讲错误码发送给手机	车机将错误码统计信息发送给手机
MSG_CMD_VIDEO_ENCODER_JPEG	0x00018056	无	HU->MD	车机发送给手机，要求传输jpeg图片代替h.264视频数据	必须在车机发送MSG_CMD_VIDEO_ENCODER_INIT之前使用
MSG_CMD_VIDEO_ENCODER_JPEG_ACK	0x00010057	无	MD->HU	手机发送给车机，确认可支持jpeg模式	可以支持的情况下回复，不支持则不回复
MSG_CMD_BT_HFP_CALL_STATUS_COVER	0x00010058	message CarlifeCallStatusCover { Required int callStatus = 1; Required String phoneNum = 2; Required String name = 3; }	MD->HU	手机发送给车机，车机需要显示一个电话状态的遮罩	当支持蓝牙 Internal UI 的时候，才有效
MSG_CMD_MENU_EXIT	0x00010059		MD->HU	当用户手动退出手机端CarLife之前发送该消息给车机端	
Reserved	0x0001005A~0x0001006F				保留 ID
MSG_CMD_HUMAN_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->MD	车机端发送给手机端的连接统计项信息 nConnectTotal 车机端或手机端发起连接请求的总次数 nSuccessCount 连接成功并显示CarLife首页的次数 nFailedCount 连接失败未成功显示CarLife首页的次数 nConnectTime 成功连接并显示首页的平均时间 ErrorType 根据上报连接失败错误码	当车机端成功连接手机时上报到手机端

表 10.2-2: CarLife 功能定制消息参数配置图

Key(String)	Value(int)	用途
VOICE_MIC	USE_VEHICLE_MIC =0; (Default) USE_MOBILE_MIC =1;	定制语音录音源，默认用车机 MIC
VOICE_WAKEUP	ENABLE = 1;(default) DISABLE = 0;	定制语音唤醒，默认唤醒可用
BLUETOOTH_AUTO_PAIR	ENABLE = 1; DISABLE = 0; (default)	定制蓝牙自动匹配，默认自动匹配
BLUETOOTH_INTERNAL_UI	ENABLE = 1; DISABLE = 0; (default)	定制蓝牙电话，默认不可用
FOCUS_UI	ENABLE = 1; DISABLE = 0; (default)	定制焦点态，默认不可用
MEDIA_SAMPLE_RATE	ORIGINAL_SAMPLE_RATE = 0;(default) CUSTOMIZED_SAMPLE_RATE_48K=1;	定制 MD 传入 HU 的 PCM 流的采样率，默认不定制
AUDIO_TRANSMISSION_MODE	INDEPENDENT_CHANNEL_MODE=0; (default) BLUE_TOOTH_MODE=1;	定制 MD 向 HU 传输音频流方式，默认为使用专用音频通道的进行音频传输
ENGINE_TYPE	FUEL_OIL=0;(default) ELECTRICITY_POWER=1;	定制运行 CarLife 的车机类型，默认为燃油车
INPUT_DISABLE	INPUT_NOTDISABLE =0; INPUT_DISABLE =1; (default)	是否根据车速屏蔽输入法，默认是只要 HU 发

		车速过来就启用屏蔽

10.2.3. 时序图

部分控制通道消息时序图如下图所示。

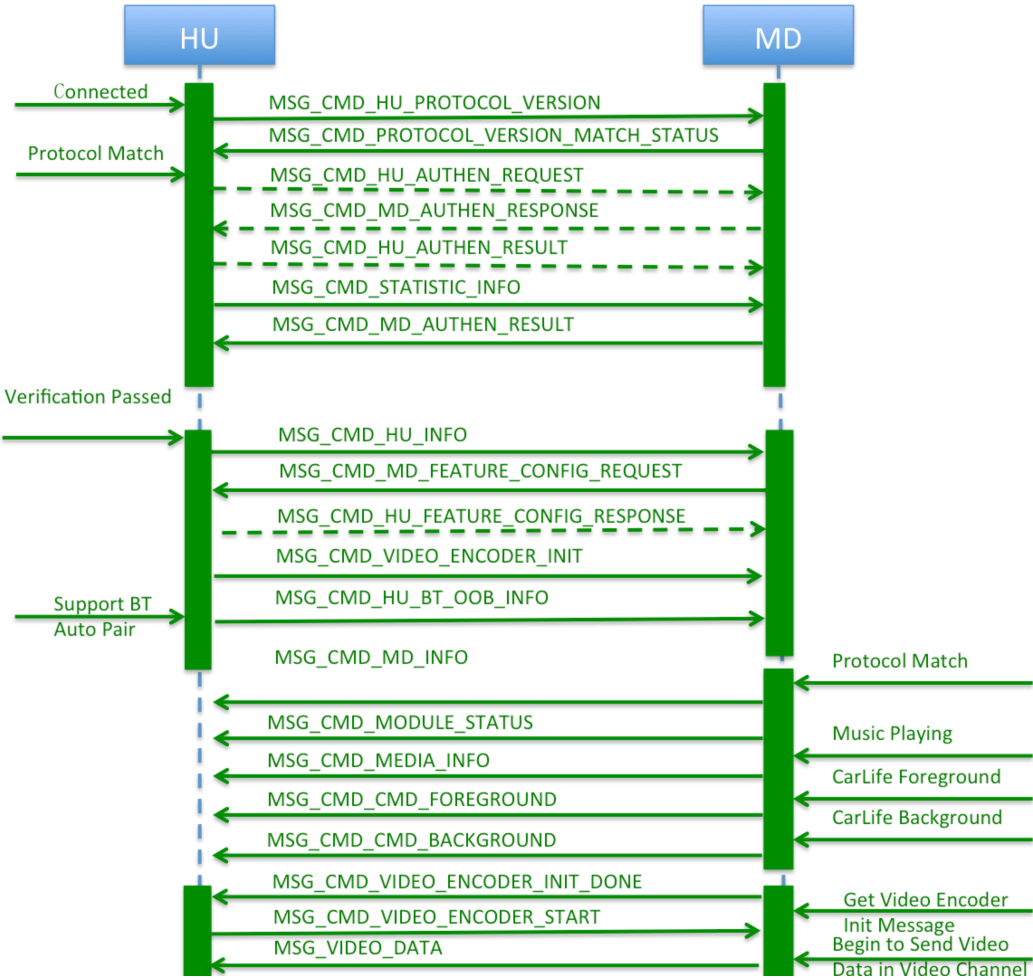


图 10.2-2：部分控制通道消息时序图

10.3. 视频消息

MD 和 HU 建立连接后，双方完成宽高参数，帧率参数等信息交换，并完成其他一些准备工作后，MD 使用 H.264 baseline profile 编码生成视频流，并通过连接时建立的 TCP socket 通道进行数据传输。HU 侧应尽量保持通道的畅通，在任何条件下，不应阻塞视频通道数据的读写；并在接收到视频数据后，尽快完成解码和显示，以减少延迟。

视频数据包的结构以及相关消息体详见小节中描述。

10.3.1. 包结构

移动设备端会将 CarLife 应用界面编码成视频流并传输到车机端，车机端需解码视频流并实时的显示在车机屏幕上。CarLife 视频消息结构如图所示。

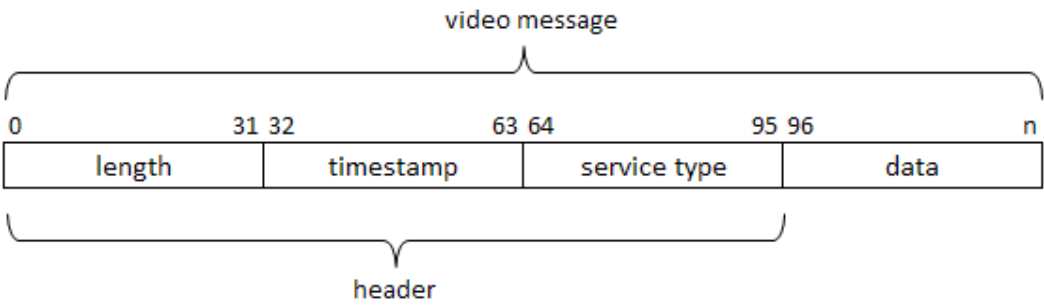


图 10.3-1： CarLife 视频消息结构

CarLife 视频消息 header 的长度固定，为 12 个字节，各字段定义如表所示。

表 10.3-1： CarLife 视频消息各字段定义

字段名称	位偏移	大小(bit)	说明
length	0~31	32	data 字段的长度
timestamp	32~63	32	视频时间戳，两个用途： 1. 用来与音频进行同步； 2. 由于移动设备会一直发送这个时间戳，可以将其作为基准时间轴用于判断音视频是否过期
service type	64~95	32	消息服务类型。 移动设备在 CarLife 处在前后台时所需传输的数据，service type 定义为以下两个： 1. CarLife 应用在前台，正在发送被编码后的视频数据：MSG_VIDEO_DATA（0x00020001）； 2. CarLife 应用在后台，发送特定结构的心跳包：MSG_VIDEO_HEATBEAT（0x00020002）。在 CarLifeV4.0 之后，由于支持手机后台截屏，所以 CarLife 应用在后台时也是发送视频数据 MSG_VIDEO_DATA

			(0x00020001)
data	96~n	length*8	编码后的视频数据字节流

10.3.2. 相关消息体

视频相关的消息体，如表所示：

表 10.3-2:

消息名称	service type	data 字段 pb 结构	流向	含义及作用	应用场景
MSG_CMD_VIDEO_ENCODER_INIT	0x00018007	<pre>message CarLifeVideoEncoderInfo { required int32 width = 1; required int32 height = 2; required int32 frameRate = 3; }</pre>	HU->MD	车机发送给移动设备的视频引擎初始化请求。	连接建立完成后，首次视频相关的交互消息。
MSG_CMD_VIDEO_ENCODER_INIT_DONE	0x00010008	<pre>message CarLifeVideoEncoderInfo { required int32 width = 1; required int32 height = 2; required int32 frameRate = 3; }</pre>	MD->HU	移动设备发送给车机的视频引擎初始化成功消息，返回 MD 最终确定的分辨率和帧率参数。	MD 侧完成视频引擎初始化后答复 HU 侧时。
MSG_CMD_VIDEO_ENCODER_START	0x00018009	无	HU->MD	车机发送给移动设备开始传输视频的消息。	HU 接收到 MSG_CMD_VIDEO_ENCODER_INIT_DONE，准备开始接收视频数据后；或 HU 由视频暂停状态，恢复至视频正常状态时。
MSG_CMD_VIDEO_ENCODER_PAUSE	0x0001800A	无	HU->MD	车机发送给移动设备暂停视频传输的消息。	HU 暂不需要显示视频，或其他原因 HU 侧视频模块进入暂停状态时。
MSG_CMD_VIDEO_ENCODER_FRAME_RATE_CHANGE	0x0001800C	<pre>message CarLifeVideoFrameRate { required int32 frameRate = 1; }</pre>	HU->MD	车机发送给移动设备的视频帧率变更请求。	HU 侧需要对最大帧率进行调整时。

RAME_RATE_CHANGE		}			
MSG_CMD_VIDEO_ENCODER_FRAME_RATE_CHANGE_DONE	0x0001000D	<pre> message CarLifeVideoFrameRate { required int32 frameRate = 1; } </pre>	MD->HU	移动设备发送给车机的视频帧率变更成功消息，返回变更后的帧率。	MD 侧完成视频最大帧率调整后，答复 HU。

各消息体中的参数字段详细解释见下面各个小节。

10.3.2.1. MSG_CMD_VIDEO_ENCODER_INIT

Field Name	Type	Optional/Required	Description
width	int32	Required	HU 显示区域的宽度（单位：像素）
height	int32	Required	HU 显示区域的高度（单位：像素）
frameRate	int32	Required	HU 希望的帧率，帧每秒

表 10.3-3

10.3.2.2. MSG_CMD_VIDEO_ENCODER_INIT_DONE

Field Name	Type	Optional/Required	Description
width	int32	Required	MD 确认的分辨率宽度（单位：像素）

height	int32	Required	MD 确认的分辨率高度（单位：像素）
frameRate	int32	Required	MD 确认的帧率，帧每秒

表 10.3-4

10.3.2.3. MSG_CMD_VIDEO_ENCODER_FRAME_RATE_CHANGE

Field Name	Type	Optional/ Required	Description
frameRate	int32	Required	HU 希望的最大帧率，帧每秒

表 10.3-5

10.3.2.4. MSG_CMD_VIDEO_ENCODER_FRAME_RATE_CHANGE_DONE

Field Name	Type	Optional/ Required	Description
frameRate	int32	Required	MD 确认的最大帧率，帧每秒

表 10.3-6

10.3.3. 时序图

以下各小节描述的消息交互过程，都是在命令消息通道中进行，与视频数据通道无关。

10.3.3.1. 建立连接时视频消息相关时序图

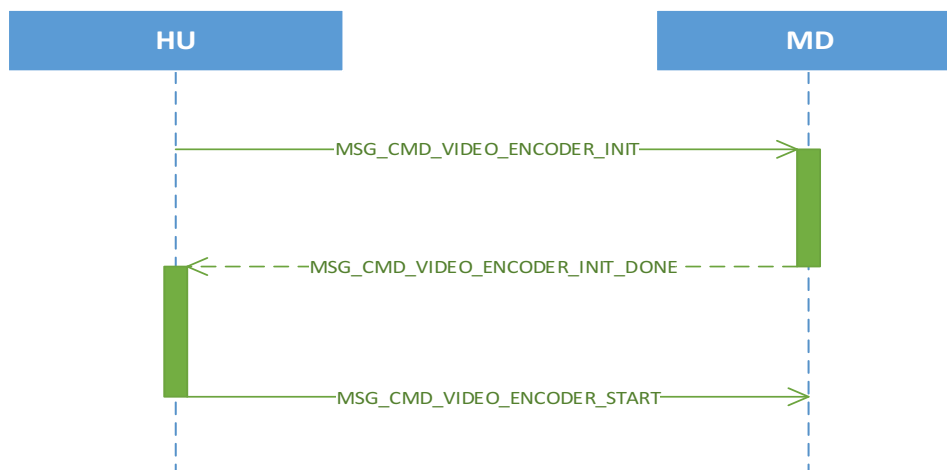


图 10.3-2

在连接建立完成后，HU 侧需要在命令消息通道，主动向 MD 侧发送初始化编码器的消息 `MSG_CMD_VIDEO_ENCODER_INIT`，消息体中包含需要的视频的宽高参数和最大帧率限制参数；之后等待 MD 侧完成编码器初始化工作后，回复初始化完成的消息 `MSG_CMD_VIDEO_ENCODER_INIT_DONE`，消息体中携带有 MD 侧最终确定的宽高参数和最大帧率限制数；HU 接收到消息后，根据其中参数完成解码器创建，及其他视频处理相关准备工作后，即可发送 `MSG_CMD_VIDEO_ENCODER_START`。此后，MD 开始产生视频数据，并通过专门为视频数据传输建立的 TCP socket 通道向 HU 发送。

10.3.3.2. 调整帧率时视频消息相关时序图

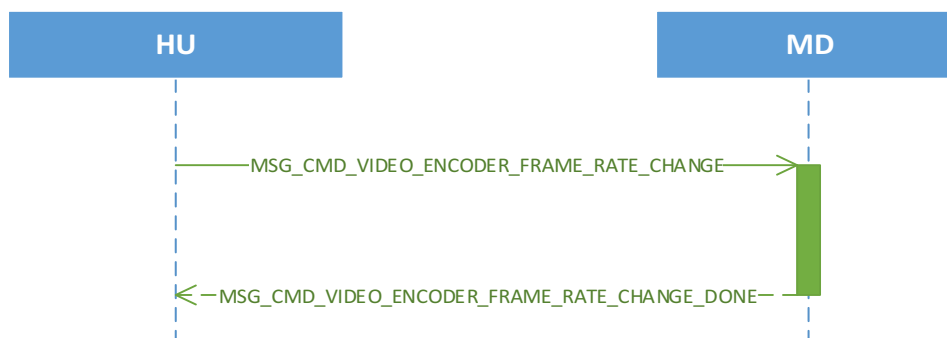


图 10.3-3

HU 侧在需要降低或升高视频帧率时，需要通过命令消息通道，发送帧率调整消息 `MSG_CMD_VIDEO_ENCODER_FRAME_RATE_CHANGE` 至 MD 侧，消息携带有目标帧率的参数；MD 侧接收到消息，校验参数合理性，并在调整完毕后，会返回相应的确认消息 `MSG_CMD_VIDEO_ENCODER_FRAME_RATE_CHANGE_DONE`，消息携带有 MD 侧最终确认的帧率调整结果。

10.3.3.3. 暂停视频数据时视频消息相关时序图

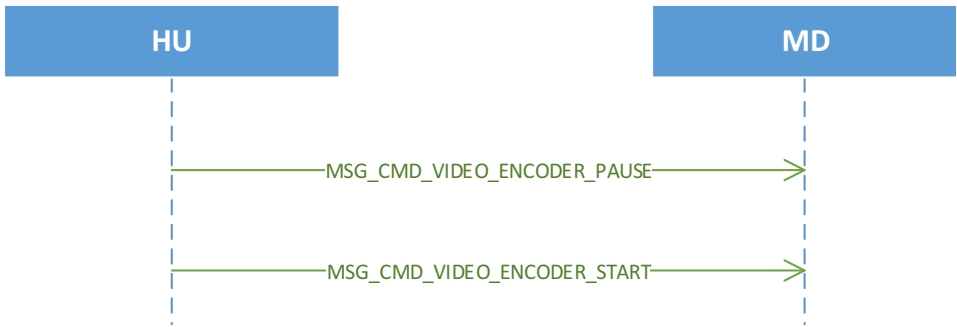


图 10.3-4

HU 侧在关闭屏幕，切换到后台运行等，无需再显示视频画面的场景中，应通知 MD 侧暂停视频数据的生成。HU 使用命令消息通道，发送 MSG_CMD_VIDEO_ENCODER_PAUSE 命令；MD 侧接收到此消息后，视频引擎会进入暂停状态，停止产生视频数据，仅发送心跳包维持连接。

HU 侧从关闭屏幕，切换到后台运行等状态中，恢复到需要显示视频画面时，需要在命令消息通道中，发送 MSG_CMD_VIDEO_ENCODER_START 命令，通知 MD 侧恢复视频引擎。

10.4. 音频消息

移动设备端会将 CarLife 应用的音频数据编码并通过相应的 TCP socket 传输到车机端，车机端需解析音频数据并决定如何播放这些音频。

10.4.1. 包结构

CarLife 音频消息结构如图 12-1 所示，

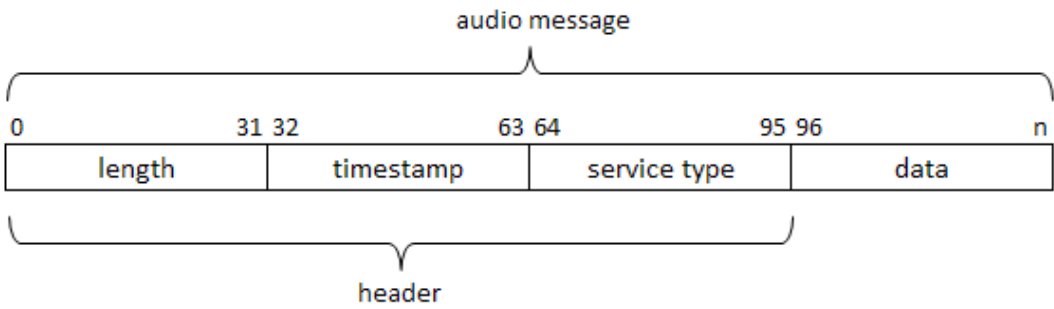


图 10.4-1: CarLife 音频消息结构

CarLife 音频消息 header 的长度固定，为 12 个字节，各字段定义如表 12-1

所示。

字段名称	Offsets	字段所占大小 (bit)	字段属性以及含义
length	0~31	32	data 字段的长度。
timestamp	32~63	32	用于与视频进行同步的时间戳信息。
service type	64~95	32	消息服务类型，移动设备传输音频数据和音频的状态。 对于 Media 类型的音频， service type 定义为以下值：初始化、数据发送、暂停、恢复播放。 对于 TTS 类型的音频， service type 定义为以下值：初始化、数据发送和停止。
data	96~n	length*8	如果 service type 是数据发送，此处是编码后的音频数据字节（小端字节序）；如果是其他 service type，此处是用 PB 序列化其参数后的字节流。

表 10.4-1: CarLife 语音数据消息各字段定义

10.4.2. 相关消息体

1. Media 音频消息定义

对于 Media 消息，除了 MSG_MEDIA_DATA 消息的 data 字段是原始的 PCM 流之外，其他消息对应的 data 字段均用 PB 序列化。移动设备端 service type 及对应的 data 字段定义如表 12-2 所示。

表 10.4-2: CarLife Media Audio 消息的 service type 和 data

字段宏定义名称	service type 值	Data	流向	含义及作用	应用场景
MSG_MEDIA_INIT	0x00030001	<pre>message CarLifeMediaInit { required int32 sampleRate = 1; required int32 channelConfig = 2; required int32 sampleFormat = 3; }</pre>	MD->HU	移动设备端 Media 的初始化，主要包括：采样率、通道配置、量化精度。	启动音乐播放时需要

MSG_MEDIA_PAUSE	0x00030003	无	MD->HU	移动设备端 Media 暂停。	暂停音乐播放时需要
MSG_MEDIA_RESUME_PLAY	0x00030004	无	MD->HU	移动设备端 Media 恢复播放。	恢复音乐播放时需要
MSG_MEDIA_DATA	0x00030006	PCM 格式的 Media 音频流	MD->HU	移动设备端发送 Media 音频数据。	音乐播放时传输音乐数据

表 12-2: MSG_MEDIA_INIT 数据部分字段定义

Field Name	Type	Optional/ Required	Description
sampleRate	int32	Required	采样率（单位 Hz）
channelConfig	int32	Required	通道数（单音轨-1 或双音轨-2）
sampleFormat	int32	Required	量化格式（单位 bits）

2. Navi TTS 音频消息定义

对于 Navi TTS 消息，除了 MSG_NAVI_TTS_DATA 消息的 data 字段是原始的 PCM 流之外，其他消息对应的 data 字段均用 PB 序列化。移动设备端 service type 及对应的 data 字段定义如表 13 所示。

表 12-3: CarLife Navi TTS Audio 消息的 service type 和 data

字段宏定义名称	service type 值	data	流向	含义及作用	应用场景
MSG_NAVI_TTS_INIT	0x00040001	message CarLifeTTSInit { required int32 sampleRate = 1; required int32 channelConfig = 2; required int32 sampleFormat = 3; }	MD->HU	移动设备端 Navi TTS 的初始化，主要包括：采样率、通道配置、量化精度。	开始播报导航 TTS 时需要
MSG_NAVI_TTS_END	0x00040002	无	MD->HU	移动设备端 Navi TTS 播放停止。	停止播报导航 TTS 时需要
MSG_NAVI_TTS_DATA	0x00040003	PCM 格式的 TTS 音频流	MD->HU	移动设备端发送 Navi TTS 音频数据。	播报导航 TTS 时传输 TTS 数据

表 12-4: MSG_NAVI_TTS_INIT 数据部分字段定义

Field Name	Type	Optional/ Required	Description
sampleRate	int32	Required	采样率（单位 Hz）
channelConfig	int32	Required	通道数（单音轨-1 或双音轨-2）

sampleFormat	int32	Required	量化格式（单位 bits）
--------------	-------	----------	---------------

10.4.3. 时序图

1. 手机端将 Navi TTS 数据发送到车机端，如图 12-2 所示：

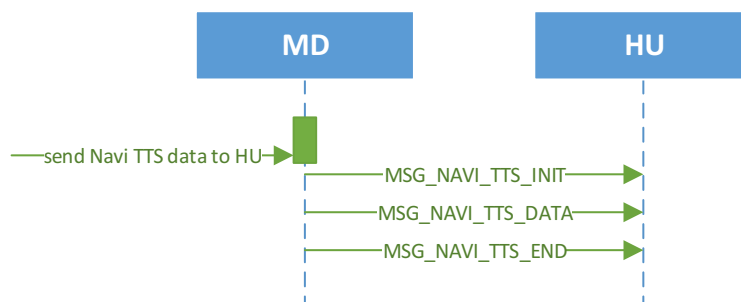


图 10.4-2：Navi TTS 通道消息时序图

其中，手机端发送 Navi TTS 数据前，首先发送包含音频初始化参数的 MSG_NAVI_TTS_INIT 消息，然后发送标示为 MSG_NAVI_TTS_DATA 的音频数据。当手机停止发送 Navi TTS 数据后，会发送 MSG_NAVI_TTS_END 通知车机。

2. 手机端将 Media 数据发送到车机端，如图 12-3 所示：

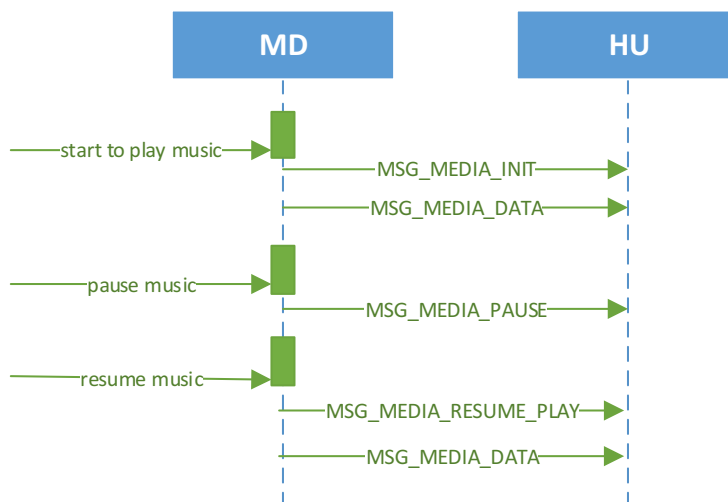


图 10.4-3：Media 通道消息时序图

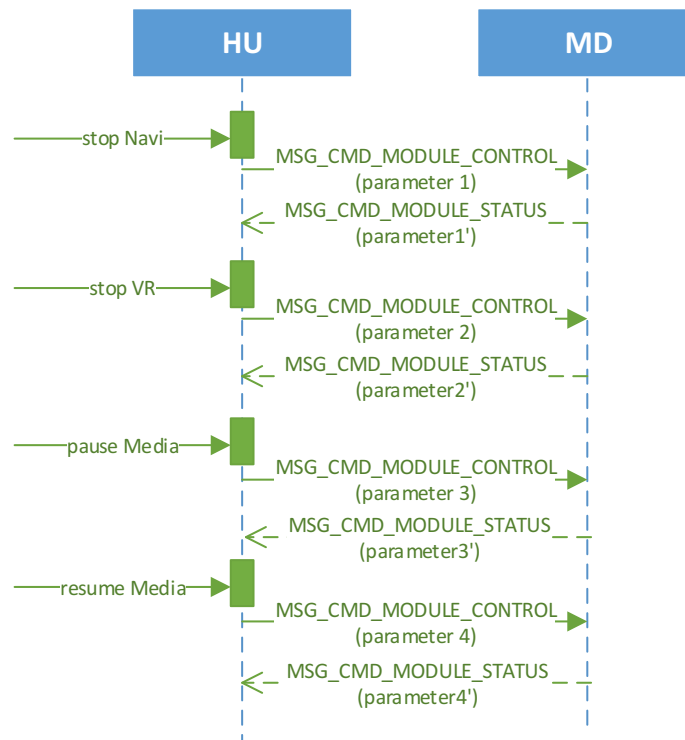
其中，

手机端音乐开始播放时，手机会先发送包含音频初始化信息的 MSG_MEDIA_INIT 消息到车机端，然后发送标识为 MSG_MEDIA_DATA 的音频数据；

当手机端音乐暂停时，手机会通过 MSG_MEDIA_PAUSE 消息通知车机端音乐暂停的状态；

当手机端音乐恢复播放时，手机端会通过 MSG_MEDIA_RESUME_PLAY 消息通知车机端音乐已恢复播放，同时，发送标识为 MSG_MEDIA_DATA 的音频数据；

3. 车机端通过命令通道控制手机端的导航停止、VR 停止以及 Media 的暂停和恢复播放，如图 12-4 所示：



10.4-4：命令通道控制音频应用时序图

其中，车机端通过 MSG_CMD_MODULE_CONROL 命令来控制手机端 Navi、VR、Media 模块的暂停（同时可控制 Media 模块恢复播放），手机端对应模块状态发生改变后，会通过 MSG_CMD_MODULE_STATUS 将模块变化的状态反馈给车机端。

10.5. 语音消息

10.5.1. 包结构

CarLife 的语音音频数据通过语音（VR）通道从移动设备传输到车机端，同时车机端将 MIC 录制到的音频数据也通过语音（VR）通道传输到移动设备端。CarLife 语音数据消息结构

如下所示。

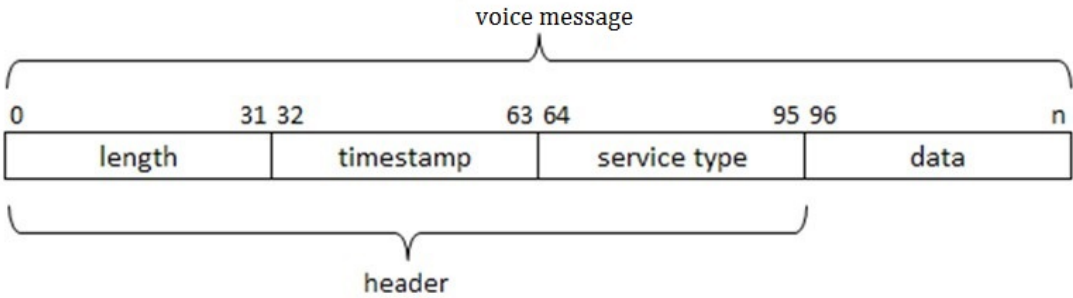


图 10.5-1 CarLife 语音消息结构

CarLife 语音数据消息 header 的长度固定，为 12 个字节，各字段定义如表 7 所示。

字段名称	位偏移	大小(bit)	说明
字段名称	Offsets	字段所占大小 (bit)	字段属性以及含义
length	0~31	32	data 字段的长度（网络字节序）。
timestamp	32~63	32	时间戳信息，可不加（网络字节序）
service type	64~95	32	语音数据信息的 service type（网络字节序）。
data	96~	建议值 1024	这是实际传输的录音数据，录音数据要求是 16Bit 位深（小端序），16K 采样率，单通道

表 10.5-1 CarLife 语音数据消息各字段定义

10.5.2. 相关消息体

表 10.5-2 是 CarLife VR 功能所需要用到的所有消息体，其中走控制通道的消息都需要经过 PB 封装，而走语音（VR）通道的语音音频数据消息则不需要，因为语音（VR）通道是使用输送数据流的方式，用 PB 封装会增加压缩和解压的时间。

表 10.5-2 CarLife 控制消息的 service type 及 data 字段定义

消息名称	service type	data	流向	含义及作用	应用场景
MSG_CMD_MIC_RECORD_WAKEUP_START	0x00010022	无	MD->HU	该消息通过命令通道传递,手机通知车机进入唤醒状态,开始录音。如果MSG_CMD_HU_FEATURE_CONFIG_RESPONSE配置使用车机MIC,则车机端收到该命令后,需将车机MIC录音数	语音唤醒启动时,由手机端发出

				据流压缩成 package 包，每包大小 1KB，逐个发送到手机端，具体发送方式请参考 MSG_VR_DATA；如果配置使用手机端 MIC，则录音功能由手机端自身完成；	
MSG_CMD_MIC_RECORDED_END	0x00010023	无	MD->HU	该消息通过命令通道传递，移动设备通知车机暂停录音。	语音唤醒或者语音识别结束时，由手机端发出
MSG_CMD_MIC_RECORD_RECOGNITION_START	0x00010024	无	MD->HU	该消息通过命令通道传递，移动设备通知车机进入语音识别状态，开始录音。如果 MSG_CMD_HU_FEATURE_CONFIG_RESPONSE 配置使用车机 MIC，则车机端收到该命令后，需将车机 MIC 录音数据流压缩成 package 包，每包大小 1KB，逐个发送到手机端，具体发送方式请参考 MSG_VR_DATA；如果配置使用手机端 MIC，则录音功能由手机端自身完成；	语音识别启动时，由手机端发出
MSG_VR_DATA	0x00058001	这是实际传输的录音数据，录音数据的质量请参看前文所述，每个数据包大小 1KB，该数据包不需要经过 PB 封装	HU->MD	该消息通过语音通道传递，车机发送给移动设备端 MIC 录制的语音 PCM 流，该消息结构就是上文包结构所阐述的。	车机端往手机端发送录音数据时，由车机端发出
MSG_VR_AUDIO_INIT	0x00050002	message CarLifeTTSInit { required int32 sampleRate = 1; required int32 channelConfig = 2; required int32	MD->HU	该消息通过语音通道传递，移动设备端 VR TTS 的初始化，主要包括：采样率、通道配置、量化精度，该消息结构就是上文包结构所阐述的。	语音识别过程中如果有语音 TTS 产生，则每次语音 TTS 播报开始时由手机端发出

		sampleFormat =3; }			
MSG_VR_AUDIO_DATA	0x00050003	PCM 格式的 VR TTS 音频流	MD->HU	该消息通过语音通道传递, 移动设备端发送 VR TTS 音频数据, 该消息结构就是上文包结构所阐述的。	语音 TTS 播报时由手机端发出, 进行实时传输数据流到车机端
MSG_VR_AUDIO_STOP	0x00050004	无	MD->HU	该消息通过语音通道传递, 移动设备端 VR TTS 播放结束时发送, 该消息结构就是上文包结构所阐述的。	标志这段语音 TTS 播报已到结尾, 由手机端发出。 注意: 此消息不是 VR TTS 打断消息, 仅代表当前 VR TTS 已到结尾, VR TTS 打断消息请参照: MSG_VR_AUDIO_INTERRUPT
MSG_CMD_MODULE_STATUS	0x00010026	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_MEDIAPCM_MODULE_ID=7; CARLIFE_EDOG_MODULE_ID=8; (P/S: PB_VERSION 2.0 新增)	MD->HU	该消息通过命令通道传递, 主要用于手机与车机互联时同步状态时需要用到: 手机通知车机音乐、语音、导航等模块状态已改变。 备注: 1、CARLIFE_MEDIAPCM_MODULE_ID 目前仅用于 iOS 设备。某些支持 iPOD 音乐播放的车机, 当车机端 CarLife 切后台 iPOD 音乐播放在前台时, 可以使用该命令暂停 MD 向 HU 传输音乐 PCM 流, 避免底层带宽不够用。 2、	在音乐、导航、语音等状态有变化时, 手机端会通过该消息通知车机端

		<pre> 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_RUNNING =1; } enum ConnectState { CONNECT_STATUS_ADB=1, CONNECT_STATUS_AOA=2, </pre>		<pre> enum ModuleId { CARLIFE_EDOG_MODULE_ID=8; CARLIFE_CRUISE_ID=9; } </pre> 这两个 MODULE_ID 为 (PB_VERSION 2.0 新增)	
--	--	--	--	--	--

		<pre> CONNECT_STATUS_ NCM_ANDROID=3, CONNECT_STATUS_ NCM_IOS =4, CONNECT_STATUS_ WIFI=5, } enum MicState { MIC_STATUS_USE_ VEHICLE_MIC =0; MIC_STATUS_USE_ MOBILE_MIC =1; } enum MediaPCMState { MEDIAPCM_STATU S_IDLE=0; MEDIAPCM_STATU S_RUNNING=1; } enum EDogState { EDOG_STATUS_IDL E =0; EDOG_STATUS_RU NNING =1; } enum CruiseState { CRUISE_ STATUS_IDLE =0; CRUISE_ STATUS_RUNNING =1; } message </pre>			
--	--	---	--	--	--

		<pre> CarlifeModuleStatusList { required int32 cnt = 1; repeated com.baidu .carlife.protobuf.CarlifeModuleStatus moduleStatus = 2; } message CarlifeModuleStatus { required int32 moduleID = 1; required int32 statusID = 2; } </pre>			
MSG_CMD_MODULE_CONTROL	0x00018028	<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_MEDIAPCM_MODULE_ID=7; CARLIFE_EDOG_MODULE_ID=8; (P/S: PB_VERSION 2.0 新增) CARLIFE_CRUISE_ID=9; } </pre>	HU->MD	该消息通过命令通道传递，主要用于手机与车机互联时同步状态时需要用到的：车机端可以通过该消息控制手机端音乐，语音，导航等模块。 备注： 1、CARLIFE_MEDIAPCM_MODULE_ID 目前仅用于 iOS 设备。某些支持 iPod 音乐播放的车机，当车机端 CarLife 切后台 iPod 音乐播放在前台时，可以使用该命令暂停 MD 向 HU 传输音乐 PCM 流，避免底层带宽不够用。 3、 <pre> enum ModuleId { CARLIFE_EDOG_MODU </pre>	车机端需要打开/关闭音乐、电话、导航时，就需要向手机端发送该消息，这也是非 android 车机实现硬按键控制的一个方式。

		(P/S: PB_VERSION 2.0 新增) <pre> } enum PhoneState { PHONE_STATUS_ID LE=0; PHONE_STATUS_IN COMING=1; PHONE_STATUS_O UTING=2; } enum NaviState { NAVI_STATUS_IDLE =0; NAVI_STATUS_RUN NING =1; } enum MusicState { MUSIC_STATUS_IDL E=0; MUSIC_STATUS_RU NNING=1; } enum VRState { VR_STATUS_ RECORD _IDLE =0; VR_STATUS_ RECORD _RUNNING =1; } enum ConnectState { CONNECT_STATUS_ ADB=1, CONNECT_STATUS_ AOA=2, CONNECT_STATUS_ NCM_ANDROID=3, </pre>		<pre> LE_ID=8; CARLIFE_CRUISE_ID=9; } </pre> 这两个 MODULE_ID 为 (PB_VERSION 2.0 新增)	
--	--	--	--	---	--

		<pre> CONNECT_STATUS_ NCM_IOS =4, CONNECT_STATUS_ WIFI=5, } enum MicState { MIC_STATUS_USE_ VEHICLE_MIC =0; MIC_STATUS_USE_ MOBILE_MIC =1; } enum MediaPCMState { MEDIAPCM_STATU S_IDLE=0; MEDIAPCM_STATU S_RUNNING=1; } enum EDogState { EDOG_STATUS_IDL E =0; EDOG_STATUS_RU NNING =1; } enum CruiseState { CRUISE_STATUS_ID LE =0; CRUISE_STATUS_R UNNING =1; } message CarlifeModuleStatu s { required int32 </pre>			
--	--	--	--	--	--

		<pre>moduleID = 1; required int32 statusID = 2; }</pre>			
MSG_VR_MODULE_STATUS	0x00050005	类似 MSG_CMD_MODULE_STATUS	MD->HU	该消息是 MSG_CMD_MODULE_STATUS 同步语音状态的替代者。 原因如下：如果语音启动时，MD 会先发送 MSG_CMD_MODULE_STATUS 告知 HU 语音启动了，紧接着发送 MSG_VR_AUDIO_INIT 消息通知播放问候语，由于这两条消息是通过不同通道发送的，高概率会出现后发送的 MSG_VR_AUDIO_INIT 反而先到达，以致某些 HU 会处理异常。 为了根本解决该问题，所以增加 MSG_VR_MODULE_STATUS，使之代替 MSG_CMD_MODULE_STATUS，该消息通过 VR 通道发送，这样就能保证时序上一致。	为了保证向上兼容，MSG_CMD_MODULE_STATUS 仍然会发给 HU，而 MSG_VR_MODULE_STATUS 只会发给指定的 HU，处理时需要二选一。
MSG_VR_AUDIO_INTERRUPT	0x00050006	无	MD->HU	语音识别的播报过程中，此时再次进入识别状态，手机端会发送该消息到车机端，车机端收到此消息时，需要停止当前语音的播报以及清空 VR TTS 的缓存。	在语音识别播放较长的 VR TTS 文本（如笑话）时，如需再次进入语音识别状态，则需要停止当前的 VR TTS 播报，手机端会先发送 MSG_VR_AUDIO_STOP 消息标志该段 VR

					TTS 已到结尾，再发送 MSG_VR_AUDIO_INTERRUPT 停止当前语音播报。
--	--	--	--	--	--

注：车机端针对语音通道消息进行处理时需要注意，在接收到语音通道消息时需要立即进行消息类型的判断，如果消息类型为 MSG_VR_AUDIO_INTERRUPT，则立即将缓冲区中还未播报的语音数据进行丢弃。

10.5.3. 时序图

10.5.3.1. 唤醒时序

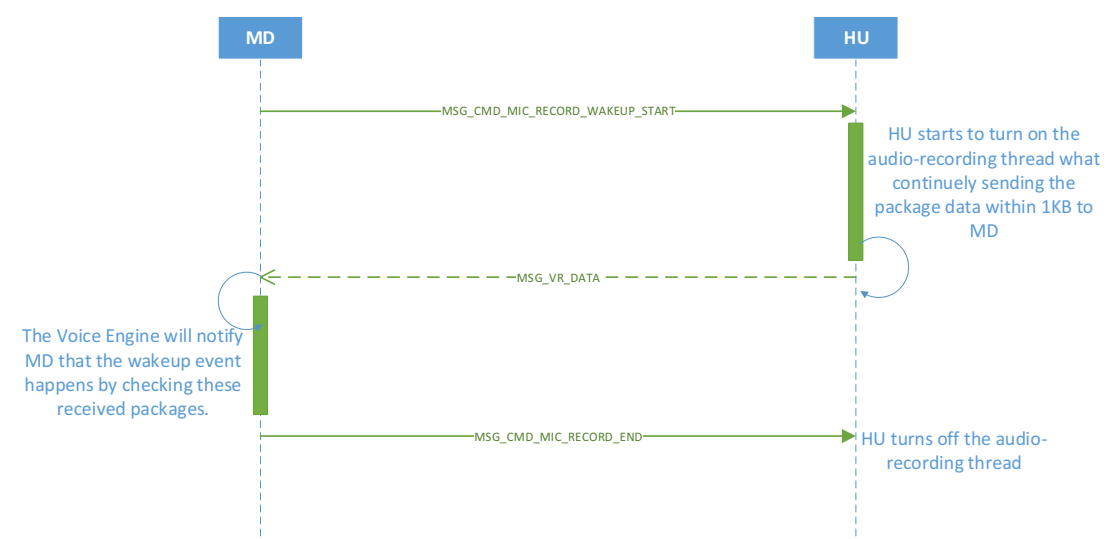


图 10.5-3-1 The Voice Wakeup Interactive Sequence Diagram

- 1) 语音唤醒启动时，手机端会先往车机端发送 `MSG_CMD_MIC_RECORD_WAKEUP_START` 消息，通知车机端语音唤醒功能开始，启动录音线程。
- 2) 车机端会持续不断地往手机端发送 `MSG_VR_DATA`，传递语音数据包，每个包大小是 1KB。
- 3) 当手机端语音引擎识别到“小度小度”时，就会结束本次唤醒，并发送 `MSG_CMD_MIC_RECORD_END` 消息通知车机端结束录音，继而开始一次语音识别过程。

10.5.3.2. 识别时序

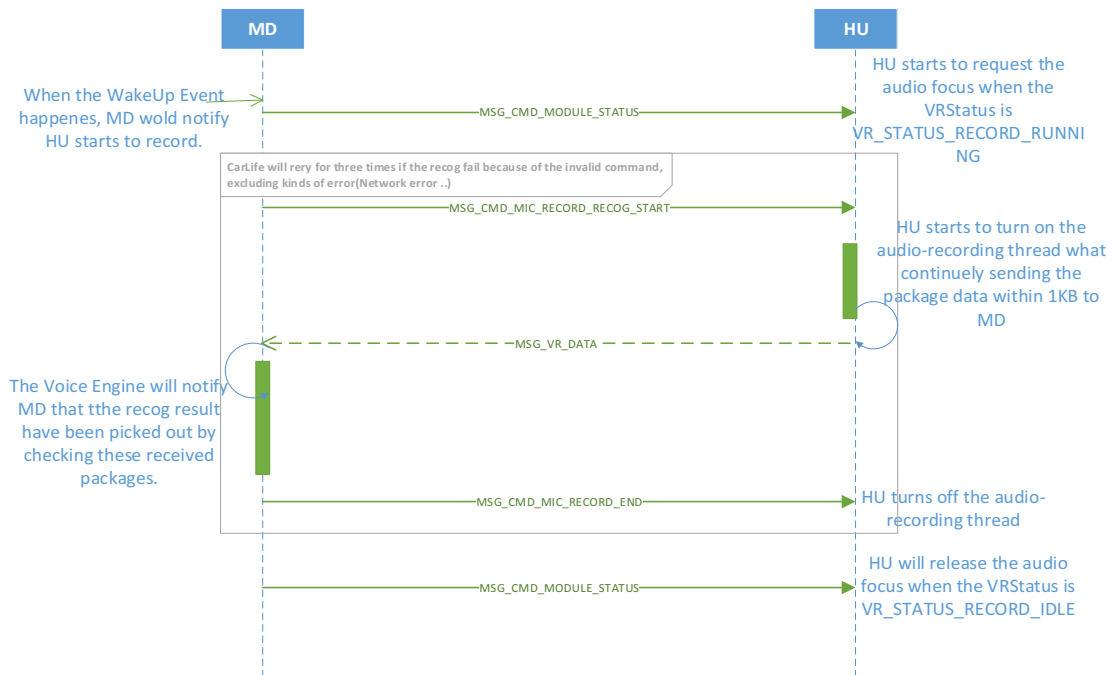


图 10.5-3-2 The Voice Recognition Interactive Sequence Diagram

- 1) 语音识别蒙层开启时，手机端会先往车机端发送 `MSG_CMD_MODULE_STATUS` 消息，通知车机端 VR 启动，车机端需要申请音频焦点,同时手机端会初始化语音识别引擎；
- 2) 语音识别引擎启动就绪后，手机端会发送 `MSG_CMD_MIC_RECORD_RECOG_START` 消息，车机端收到该消息后开启 MIC 录音线程，然后车机端应持续不断地往手机端发送 `MSG_VR_DATA` 消息，用于传递车机 MIC 录音数据包，每个包大小是 1KB；
- 3) 当手机端语音识别引擎有识别结果返回时，手机端将传输 `MSG_CMD_MIC_RECORD_END` 消息，告知车机端此时应停止 MIC 录音线程；
- 4) 在语音识别蒙层已打开的情况下，重新开启新一轮语音识别时，手机端只会重发 `MSG_CMD_MIC_RECORD_RECOG_START` 消息，不会再次发送 `MSG_CMD_MODULE_STATUS` 消息（参数值 `VR_STATUS_RECORD_RUNNING`）。
备注：`MSG_CMD_MODULE_STATUS` 消息（参数值 `VR_STATUS_RECORD_RUNNING`）只是在语音识别蒙层打开时发送；
- 5) 当语音识别蒙层关闭时，手机端会停止语音识别引擎，同时发送 `MSG_CMD_MODULE_STATUS`（参数值 `VR_STATUS_RECORD_IDLE`）给车机端，收到该消息后车机端需要释放音频焦点。另外，手机端也会发送 `MSG_CMD_MIC_RECORD_END` 消息，车机端按需终止录音线程，保证不会再给手机端传输 `MSG_VR_DATA` 消息。

10.5.3.3. 录音源定制时序

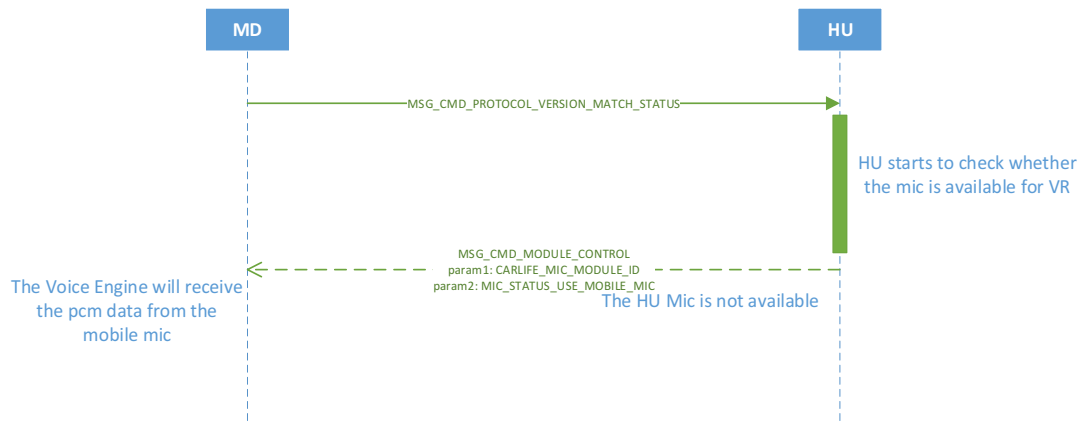


图 10.5-3-3 MIC 定制交互图

语音识别和语音唤醒功能均可使用车机 MIC 和手机 MIC 完成，如无特殊说明，默认使用车机 MIC 来完成。

如果车机端 MIC 的录音质量不能满足语音引擎要求，而考虑使用手机自带 MIC 的话，实现方式有两种，建议使用第一种方式统一完成配置：

- (1) 当车机端向手机端发送 `MSG_CMD_HU_FEATURE_CONFIG_RESPONSE` 消息时，`VOICE_MIC` 参数值配置为 `USE_MOBILE_MIC`，具体请参考 `MSG_CMD_HU_FEATURE_CONFIG_RESPONSE` 消息的定义；
- (2) 可以发送 `MSG_CMD_MODULE_CONTROL` 消息，`ModuleId` 参数值是 `CARLIFE_MIC_MODULE_ID`，`MicState` 参数值是 `MIC_STATUS_USE_MOBILE_MIC`，发送的时机是车机端收到 `MSG_CMD_PROTOCOL_VERSION_MATCH_STATUS` 后。

10.5.3.4. VR-TTS 播报时序

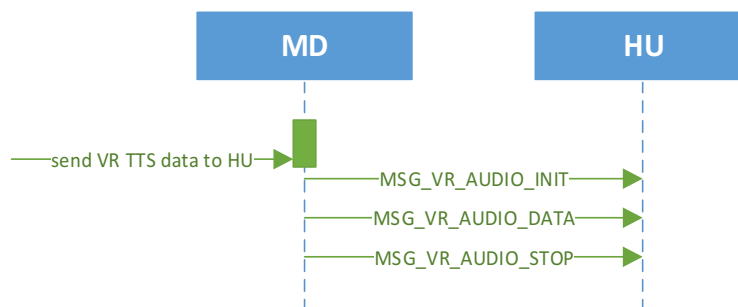


图 10.5-3-4 语音 TTS 播报时序图

- 1) 语音识别时会产生一些 VR TTS 给予用户听觉上的反馈，在手机端发送这些 VR TTS 数据前，都会先发送包含音频初始化参数的 `MSG_VR_AUDIO_INIT` 消息；
- 2) 然后手机端会通过语音通道实时不断地向车机端发送标识为

MSG_VR_AUDIO_DATA 的 VR TTS 音频数据；

- 3) 当 VR TTS 播放完毕或者 VR TTS 被打断时，均会发送 MSG_VR_AUDIO_STOP 消息通知车机端，标志这段 VR TTS 已经结束。
- 4) 如果正在播放较长的 VR TTS（如笑话、百科等）时想要重新开启新一轮语音识别，则手机端会先发送 MSG_VR_AUDIO_STOP 消息告知此段 VR TTS 已到结尾，再发送 MSG_VR_AUDIO_INTERRUPT 消息告知车机端需要停止 VR TTS 播放，然后进入 10.5.3.2 识别时序的步骤 2 和步骤 3；
此处需注意的是：如果车机端 VR TTS 还未完全停止播放就开启录音线程，容易将部分 VR TTS 录入然后当成录音数据传递给手机端造成误识别；
- 5) 在关闭语音识别蒙层时，手机端也会发送一次 MSG_VR_AUDIO_STOP 消息和 MSG_VR_AUDIO_INTERRUPT 消息，车机端可按需处理，从而保证 VR TTS 已处理完毕。

10.6. 反控消息

10.6.1. 各种反控事件处理机制

10.6.1.1. 触摸屏

如果车机的显示界面为触摸屏，移动设备同时支持透传和非透传两种反控方式。由于透传方式更直接响应更及时，所以要求使用透传的方式。

10.6.1.1.1. 透传方式

触屏事件由 ACTION_DOWN、ACTION_UP、ACTION_MOVE 三种基本的事件组成。车机端屏幕捕捉用户的触摸事件，然后把获取的这三种基本事件传送到手机端，由手机端进行处理和识别。此即为透传方式，事件顺序如下：

- 单击事件：ACTION_DOWN -> ACTION_UP
- 双击事件：ACTION_DOWN -> ACTION_UP -> ACTION_DOWN -> ACTION_UP
- 长按然后松开事件：ACTION_DOWN -> ACTION_MOVE -> ACTION_MOVE -> ... -> ACTION_UP
- 长按然后拖动事件：ACTION_DOWN -> ACTION_MOVE -> ACTION_MOVE -> ... -> ACTION_UP
- 拖动事件：ACTION_DOWN -> ACTION_MOVE -> ACTION_MOVE -> ... -> ACTION_UP

10.6.1.1.2. 坐标换算

对于触摸屏，移动设备和车机的屏幕分辨率不相同，传输的视频分辨率根据车机和移动设备分辨率的不同也不一样，因此需要将车机端触摸事件的坐标转换成移动设备中对应的坐标。转换规则如下：

(1) 车机端获取的坐标为 (x_0, y_0) ，车机端显示视频的区域大小为 (w_0, h_0) ，当接受到 MSG_CMD_VIDEO_ENCODER_INIT_DONE 消息以后获取视频流的分辨率为 (w_1, h_1) ，则传输的每个事件的 pb 结构中的坐标 (x_1, y_1) ，计算公式如下：

$$x_1 = (\text{int}) (x_0 * w_1 / w_0);$$

$$y_1 = (\text{int}) (y_0 * h_1 / h_0);$$

(2) 移动设备机端事件实际坐标为 (x_2, y_2) ，视频流的分辨率为 (w_1, h_1) ，移动设备的分辨率为 (w_2, h_2) 计算公式如下：

$$x_2 = (\text{int}) (x_1 * w_2 / w_1);$$

$$y_2 = (\text{int}) (y_1 * h_2 / h_1);$$

10.6.1.2. 硬按键

为了方便用户通过 HU 上的硬按键控制移动设备应用，提高用户体验，在反控通道增加一条车机传给移动设备的硬按键消息。根据硬按键所处位置不同，这些硬按键使用的场景大致有车机、方向盘、旋钮三种，并且车机端 CarLife 应用在前台时才能生效。车机端可以选择性支持某一些按键，如果车机端获取到某一个按键事件以后，将其在下表中所对应的 keycode 数值封装进 PB 结构中，将该消息发送给移动设备端即可。下表为具体的 keycode 值以及所代表的意义。

表 10.6-1: CarLife 硬按键消息

硬按键名称	键值	描述	备注	场景
KEYCODE_Home	0x0000 0001	Home 键	相当于点击了手机的 Home 键	车机
KEYCODE_Phone_Call	0x0000 0002	打电话	使用移动设备端 CarLife 应用拨打或者接听电话	车机、方向盘
KEYCODE_Phone_End	0x0000 0003	挂断电话	使用移动设备端 CarLife 应用挂断电话	车机、方向盘
KEYCODE_Phone_End_Mute	0x0000 0004	挂断电话，静音多媒体	使用移动设备端 CarLife 应用挂断电话，并静音多媒体	已废弃

KEYCODE_HFP	0x0000 0005	切换免提电话和移动设备电话	切换免提电话和移动设备电话	车机
KEYCODE_Selector_Next	0x0000 0006	切换到下一个焦点	在当前小区域中切换到下一个焦点	旋钮
KEYCODE_Selector_Previous	0x0000 0007	切换到上一个焦点	在当前小区域中切换到上一个焦点	旋钮
KEYCODE_Setting	0x0000 0008	进入系统设置界面	进入移动设备端 CarLife 应用的设置界面	已废弃
KEYCODE_Media	0x0000 0009	切换到多媒体界面	进入移动设备端 CarLife 应用的音乐播放界面	车机
KEYCODE_Radio	0x0000 000A	切换到 Radio 界面	进入移动设备端 CarLife 应用的 Radio 界面	已废弃
KEYCODE_Navi	0x0000 000B	切换到导航界面	进入移动设备端 CarLife 应用的导航界面	车机
KEYCODE_SRC	0x0000 000C	切换 Radio 和多媒体	切换移动设备端 CarLife 应用的音乐和电台	已废弃
KEYCODE_Mode	0x0000 000D	多媒体切换	切换移动设备端 CarLife 应用的多媒体播放源	已废弃
KEYCODE_Back	0x0000 000E	切换到上一状态	返回到移动设备端 CarLife 应用的上一个界面	车机
KEYCODE_Seek_Sub	0x0000 000F	上一首	播放上一首音乐	车机
KEYCODE_Seek_Add	0x0000 0010	下一首	播放下一首音乐	车机
KEYCODE_Volume_Sub	0x0000 0011	降低音量	降低移动设备端音量	已废弃
KEYCODE_Volume_Add	0x0000 0012	增大音量	增大移动设备端音量	已废弃
KEYCODE_Mute	0x0000 0013	静音	电话过程中静音	车机, 用于 mute 电话

KEYCODE_OK	0x0000 0014	确认键	选择焦点以后，按下该键确认点击	车机
KEYCODE_Move_Left	0x0000 0015	焦点区域左移	切换到当前小区域的左边小区域中，焦点将切换到该小区域记录的最后一个焦点	旋钮
KEYCODE_Move_Right	0x0000 0016	焦点区域右移	切换到当前小区域的右边小区域中，焦点将切换到该小区域记录的最后一个焦点	旋钮
KEYCODE_Move_Up	0x0000 0017	焦点区域上移	切换到当前小区域的上边小区域中，焦点将切换到该小区域记录的最后一个焦点	旋钮
KEYCODE_Move_Down	0x0000 0018	焦点区域下移	切换到当前小区域的下边小区域中，焦点将切换到该小区域记录的最后一个焦点	旋钮
KEYCODE_Move_Up_Left	0x0000 0019	左上移动	只在地图底图界面支持	旋钮
KEYCODE_Move_Up_Right	0x0000 001A	右上移动	只在地图底图界面支持	旋钮
KEYCODE_Move_Down_Left	0x0000 001B	左下移动	只在地图底图界面支持	旋钮
KEYCODE_Move_Down_Right	0x0000 001C	右下移动	只在地图底图界面支持	旋钮
KEYCODE_TEL	0x0000 001D	切换到电话界面	进入移动设备端 CarLife 应用的电话界面	车机
KEYCODE_MAIN	0x0000 001E	切换到主页面	进入移动设备端 CarLife 应用的主界面（第一页）	车机
KEYCODE_Media_Start	0x0000 001F	播放音乐	播放当前音乐	车机
KEYCODE_Media_Stop	0x0000 0020	暂停音乐	暂停当前音乐	车机
KEYCODE_VR_Start	0x0000 0021	打开语音	打开语音，显示语音识别界面	车机、方向盘（可长按）

KEYCODE_VR_Stop	0x0000 0022	关闭语音	关闭语音，关闭语音识别界面	车机、方向盘（可长按）
KEYCODE_NUMBER_0	0x0000 0023	数字键 0	拨打电话时的数字键	车机
KEYCODE_NUMBER_1	0x0000 0024	数字键 1	拨打电话时的数字键	车机
KEYCODE_NUMBER_2	0x0000 0025	数字键 2	拨打电话时的数字键	车机
KEYCODE_NUMBER_3	0x0000 0026	数字键 3	拨打电话时的数字键	车机
KEYCODE_NUMBER_4	0x0000 0027	数字键 4	拨打电话时的数字键	车机
KEYCODE_NUMBER_5	0x0000 0028	数字键 5	拨打电话时的数字键	车机
KEYCODE_NUMBER_6	0x0000 0029	数字键 6	拨打电话时的数字键	车机
KEYCODE_NUMBER_7	0x0000 002A	数字键 7	拨打电话时的数字键	车机
KEYCODE_NUMBER_8	0x0000 002B	数字键 8	拨打电话时的数字键	车机
KEYCODE_NUMBER_9	0x0000 002C	数字键 9	拨打电话时的数字键	车机
KEYCODE_NUMBER_10	0x0000 002D	字符键*	拨打电话时的数字键	车机
KEYCODE_NUMBER_11	0x0000 002E	字符键#	拨打电话时的数字键	车机
KEYCODE_NUMBER_DELETE	0x0000 002F	删除键	删除拨打电话输入框的一个字符	车机
KEYCODE_NUMBER_CLEAR	0x0000 0030	清除键	清除拨打电话输入框的所有字符	车机

KEYCODE_NUMBER_PL US	0x0000 0031	+键	长按*键时，发送该 Keycode	车机
-------------------------	----------------	----	-------------------	----

10.6.1.2.1. 地图界面硬按键

地图主界面，新增拖动模式按钮，若连接后识别到时物理旋钮车机,则出现该拖动模式按钮。点击该按钮进入拖动模式后,地图主界面其他元素隐藏(全屏)，光标变成游标，始终位于底图中心，用户可集中操作拖动地图和放大缩小地图。推动 knob 可拖动底图，左旋缩小地图，右旋放大地图，其他操作如单击 knob 暂不响应。用户若进入拖动模式后，选择跳转其他页面，再次返回地图页将退出拖动模式。按车机上的 BACK 可退出拖动模式，焦点回到地图页默认附近 icon。

10.6.2. 包结构

CarLife 的反控消息通过 Touch 通道来传输，主要传输车机端获取的反控事件指令到移动设备端。CarLife 反控消息结构同控制消息结构，如下图所示。

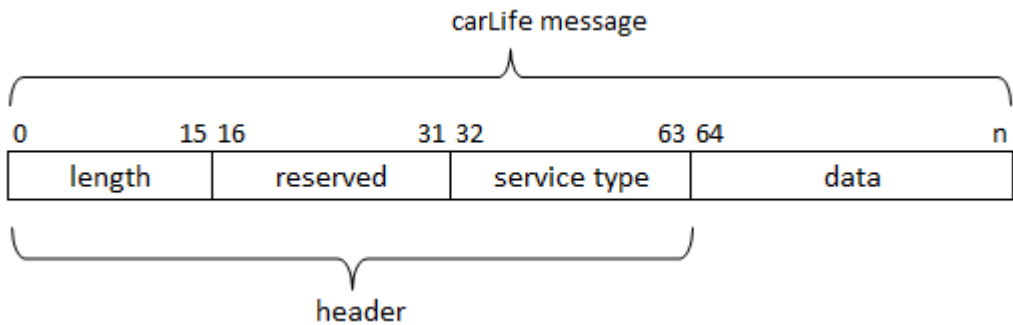


图 10.6-1： CarLife 反控消息结构

CarLife 反控消息 header 的长度固定，为 8 个字节，各字段定义如下表所示。

表 10.6-2： CarLife 反控消息各字段定义

字段名称	位偏移	大小(bit)	说明
length	0~15	16	data 字段用 PB 序列化后的长度
reserved	16~31	16	保留字段
service type	32~63	32	消息服务类型
data	64~n	length*8	service type 字段的参数，用 PB 序列化后的字节流

10.6.3. 相关消息体

反控消息的所有 service type 对应的 data 字段均用 PB 序列化。车机端和移动设备端 service type 及对应的 data 字段定义如下表所示。

表 10.6-3: CarLife 反控消息的 service type 及 data 字段定义

MSG_TOUCH_ACTION	0x00 0680 01	<pre>message CarLifeTouchAction { required int32 action = 1; required int32 x = 2; required int32 y = 3; }</pre>	车机端发送给移动设备的触摸事件。其中 action 表示事件的类型（ACTION_DOWN、ACTION_UP、ACTION_MOVE 分别用 0、1、2），其中 x 和 y 的计算详见 坐标换算 章节。（透传方式都用这个消息）
MSG_TOUCH_ACTION_DOWN	0x00 0680 02	<pre>message CarLifeTouchSinglePoint { required int32 x = 1; required int32 y = 2; }</pre>	车机端发送给移动设备的 ACTION_DOWN 事件。其中 x 和 y 的计算详见 坐标换算 章节。
MSG_TOUCH_ACTION_UP	0x00 0680 03	同上	车机端发送给移动设备的 ACTION_UP 事件。
MSG_TOUCH_ACTION_MOVE	0x00 0680 04	同上	车机端发送给移动设备的 ACTION_MOVE 事件。
MSG_TOUCH_SINGLE_CLICK	0x00 0680 05	同上	车机端发送给移动设备的单击事件。
MSG_TOUCH_DOUBLE_CLICK	0x00 0680 06	同上	车机端发送给移动设备的双击事件。
MSG_TOUCH_LONG_PRESS	0x00 0680 07	同上	车机端发送给移动设备的长按事件。
MSG_TOUCH_CAR_HARD_KEY_CODE	0x00 0680 08	<pre>message CarLifeCarHardKeyCode { required int32 keycode = 1; }</pre>	车机发送给移动设备硬按键事件，keycode 为具体事件的编号，详见 硬按键 章节。

10.6.4. 时序图

触摸屏透传方式的时序图如下：

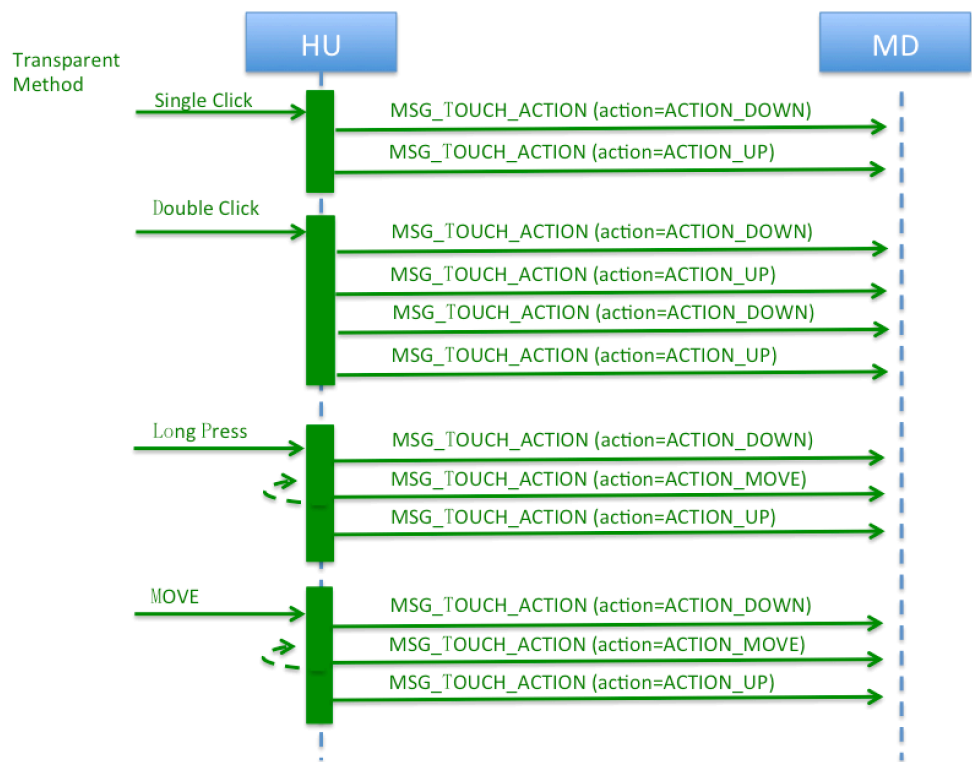


图 10.6-2：触摸屏透传方式的时序图
触摸屏非透传方式的时序图如下：

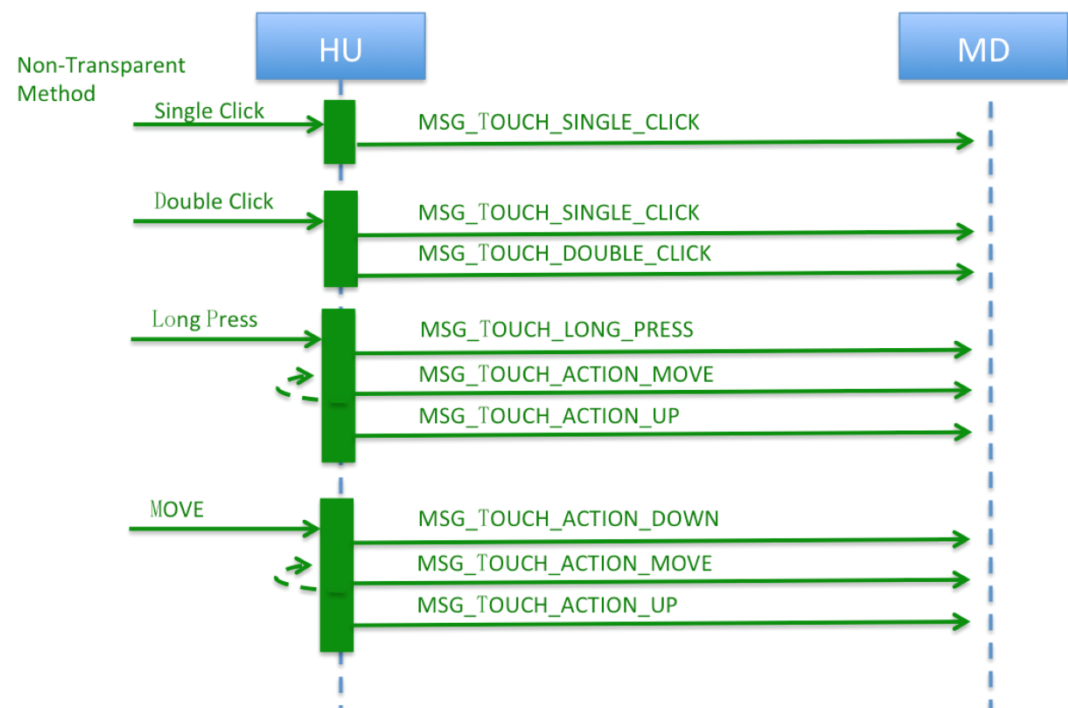


图 10.6-3：触摸屏非透传方式的时序图
物理硬按键的时序图如下：

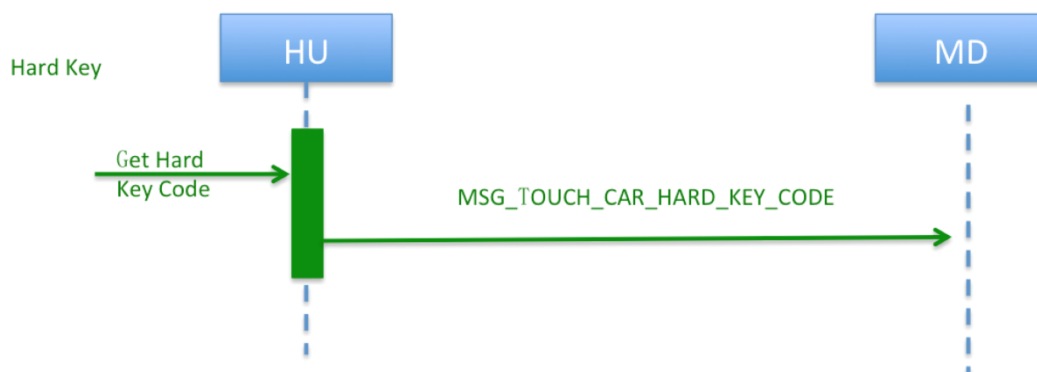


图 10.6-4: 物理硬按键的时序图

11. 手机端特殊场景策略

本章节描述手机端 **CarLife** 在锁屏、后台或者电话通话过程中时手机端跟车机端的消息交互，以及车机端该如何处理。

11.1. 手机端锁屏或者前后台切换

CarLife 新的视频方案（应用内截屏方案）对应的版本（**Android: CarLife 3.0** 及以上，**iOS: CarLife 4.0** 及以上）支持手机切后台或者锁屏，所以正常情况下 **iOS** 和安卓手机端都不会再发送前后台切换的命令。

但是由于 **iOS** 系统限制，某些情况下后台的截屏编码可能会出错，这种情况下，手机端会发送 **MSG_CMD_BACKGROUND** 消息，并停止发送视频流数据包，开始发送视频心跳包；车机端收到 **MSG_CMD_BACKGROUND** 后，必须要切换到 **CarLife** 后台运行时的提示页面，该页面具体样式参考车机端 [HMI 设计规范简述](#) 章节。

手机端 **CarLife** 从后台切换到前台时，会停止发送视频心跳包，恢复视频流数据包的传输，并通过命令通道给车机端发送协议消息：**MSG_CMD_FOREGROUND**，车机端收到该消息后从后台提示页面切换到视频页面，并重新开始视频解码播放。

对于 **Android** 不支持后台截屏的版本，处理方式也是类似的。

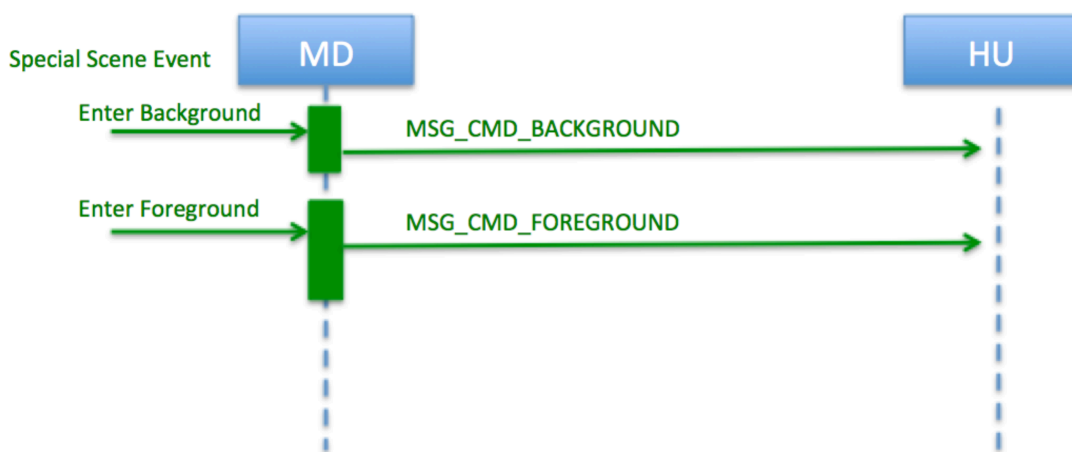


图 11.1-1

11.2. 手机端电话通话过程

如果车机端支持 CarLife 的应用内蓝牙电话功能并且手机和车机蓝牙已经配对连上，CarLife 联机使用的电话形态，请参考[蓝牙电话](#)章节。

当车机不支持 CarLife 的应用内蓝牙电话功能或者手机和车机蓝牙未连接时，手机端 CarLife 会把相关的通话状态告知车机端，以便车机端通过电话状态提示页面做出相应提示。电话状态提示页面的设计请参考[车机端 HMI 设计规范简述](#)。

当手机端电话状态发生变化时，手机 APP 会通过发送 MSG_CMD_MODULE_STATUS 消息来告诉车机当前电话的状态，此时，该消息中的 moduleID 为电话模块，statusID 分别是：

- 当通话结束时，为 PHONE_STATUS_IDLE，车机端收到该消息后从电话状态提示页面切换回到视频播放页面
- 当有新的来电时，为 PHONE_STATUS_INCOMING，车机端收到该消息后切换到电话状态提示页面，显示电话来电中
- 当呼叫中，或者通话中为 PHONE_STATUS_OUTGOING，车机端收到该消息后切换到电话状态提示页面，显示电话通话中

12. 车载数据使用及建议

CarLife 在车机端，可以获取车载相关数据并回传，弥补手机端数据不完备之处，完善用户体验。目前支持的从车机端获取的数据包括：车速，GPS 定位信息，陀螺仪加速信息，油耗。数据回传遵循数据订阅机制，传递传递消息时使用的相关通信协议详见下表。

车载数据相关的消息体，如表所示：

表 12-1: CarLife 车载数据消息的 service type 及 data 字段定义

消息名称	service type	data	流向	含义及作用	应用场景
MSG_CMD_CAR_VELOCITY	0x0001800F	<pre> message CarLifeCarSpeed { required uint32 speed = 1; optional uint64 timeStamp = 2; } </pre>	HU->MD	车机发送给移动设备的当前速度信息。	CarLife 导航中使用。
MSG_CMD_CAR_GPS	0x00018010	<pre> message CarLifeCarGps { 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>	HU->MD	车机发送给移动设备的当前位置信息。支持 WGS84 和 GCJ02 坐标系。 默认发送频率为 1HZ	CarLife 导航中使用。
MSG_CMD_CAR_GYROSCOPE	0x00018011	<pre> enum GyroType { SINGLE_AXIS_GYRO = 1; THREE_AXIS_GYRO = 2; } message CarLifeGyroscope { required GyroType gyroType = 1; required </pre>	HU->MD	车机发送给移动设备的当前汽车陀螺仪信息。 当车机采用单轴陀螺仪时，将 gyroX 设置为单轴陀螺仪的值，gyroY 和 gyroZ 赋值为 0。 陀螺仪的值为角速度，单位为度/秒。 gyroX: 车机 Y 轴水	CarLife 导航中使用。

		<pre>doublegyroX = 2; required doublegyroY = 3; required doublegyroZ = 4; optional uint64 timeStamp = 5; }</pre>		平面上的投影与正北方向的夹角。 默认发送频率为 10HZ	
MSG_CMD_CAR_ACCELERATION	0x00018012	<pre>message CarLifeAcceleration { required doubleaccX = 1; required doubleaccY = 2; required doubleaccZ = 3; optional uint64 timeStamp = 4; }</pre>	HU->MD	车机发送给移动设备的当前汽车加速度信息。 默认发送频率为 10HZ	CarLife 导航中使用。
MSG_CMD_CAR_OIL	0x00018013	<pre>message CarLifeOil { required int level =1; required int range=2; required bool lowFuleWarning = 3; }</pre>	HU->MD	车机发送给移动设备的当前汽车油耗信息。。	CarLife 通过车身油耗信息提示用户加油。
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->MD	通过档位信息标示倒车信号。 目前车机只发送 REVERSE 档位; 非倒车时发送 DRIVE 档位。	CarLife 使用该消息用于区分用户当前行驶状态, 以此禁用或激活某些功能。
MSG_CMD_CAR_DATA_SUBSCRIBE	0x00010031		MD->HU	移动设备发送给车机的订阅车身数据的消息, 消息内容为空。(查看该车机支持的数据类型, 以及各类型数据产生的频率)	CarLife 订阅车身信息
MSG_CMD_CAR_DATA_SUBSCRIBE_DONE	0x00018032	<pre>enum ModuleID { CAR_DATA_GPS=0; CAR_DATA_VELOCITY=1; CAR_DATA_GYROSCOPE=2; CAR_DATA_ACCELE</pre>	HU->MD	车机针对移动设备订阅车身数据的反馈信息。 supportFlag: 该车机是否支持当前车身数据。(1 为支持, 0 为不支持; 对于	CarLife 订阅车身信息

		<pre> RATION=3; CAR_DATA_GEAR=4; CAR_DATA_OIL=5; } message CarLifeVehicleInfoList { required int32 cnt = 1; repeatedcom.baidu.carli fe.protobuf.CarLifeVehic leInfovehicleInfo = 2; } message CarLifeVehicleInfo { required int32 moduleID = 1; required int32 supportFlag = 2; required uint32 frequency= 3; } </pre>		GPS_Module, 1 为 WGS-84 坐标系, 2 为 GCJ-02 坐标系) Frequency: 车机产生该类型数据的频率。(当不支持时设置频率为 0)	
MSG_CMD _CAR_DATA _SUBSCRIB E_START	0x00 0100 33	<pre> enum ModuleID { CAR_DATA_GPS=0; CAR_DATA_VELOCIT Y=1; CAR_DATA_GYROSC OPE=2; CAR_DATA_ACCELE RATION=3; CAR_DATA_GEAR=4; CAR_DATA_OIL=5; } message CarLifeVehicleInfo { required int32 moduleID = 1; required int32 supportFlag = 2; required uint32 frequency= 3; } </pre>	MD-> HU	移动设备要求车机开始发送订阅的车身信息, 如移动设备需要订阅多个类型数据时, 则会多次发送该消息。消息内只包含一个 CarLifeVehicleInfo。	CarLife 订阅车 身信息

MSG_CMD _CAR_DATA _SUBSCRIB E_STOP	0x00 0100 34	<pre>enum ModuleID { CAR_DATA_GPS=0; CAR_DATA_VELOCIT Y=1; CAR_DATA_GYROSC OPE=2; CAR_DATA_ACCELE RATION=3; CAR_DATA_GEAR=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-> HU	移动设备要求车机停止发送订阅的车身信息，如移动设备需要取消订阅多个类型数据时，则会多次发送该消息。消息内只包含一个 CarLifeVehicleInfo。	CarLife 订阅车身信息
---	--------------------	--	------------	---	----------------

消息中，各个字段的详细含义，在下列各小节中记述。

12.1. SG_CMD_CAR_VELOCITY

MSG_CMD_CAR_VELOCITY 消息字段定义：

Field Name	Type	Optional/ Required	Description
speed	int32	Required	当前车速，单位 km/h
timeStamp	int64	optional	此次消息在车机端发出时的时间戳，单位 ms

12.2. MSG_CMD_CAR_GPS

MSG_CMD_CAR_GPS 消息字段定义：

Field Name	Type	Optional/ Required	Description
antennaState	int32	Required	GPS 模块处于何种状态，-1 不可用，

			1 可用;
latitude	int32	Required	纬度信息, 单位 1/1,000,000 degrees
longitude	int32	Required	经度信息, 单位 1/1,000,000 degrees
height	int32	Required	海拔信息, 单位 1/10 m
speed	int32	Required	速度信息, 单位 1/100m/s
heading	int32	Required	车身朝向, 单位 1/10 degrees
pdop	int32	Required	位置精度因子, 1/10 精度单位
satsUsed	int32	Required	定位使用的卫星数
statsVisible	int32	Required	搜索到的卫星数
timeStamp	int64	Required	此次消息在车机端发出时的时间戳, 单位 ms

12.3. MSG_CMD_CAR_GYROSCOPE

MSG_CMD_CAR_GYROSCOPE 消息字段定义:

Field Name	Type	Optional/ Required	Description
gyroType	GyroType	Required	enum GyroType { SINGLE_AXIS_GYRO = 1; THREE_AXIS_GYRO = 2; } 陀螺仪的类型, 当车机采用单轴陀螺仪时, 将 gyroX 设置为单轴陀螺仪的值, gyroY 和 gyroZ 赋值为 0
gyroX	int32	Required	车机 Y 轴水平面上的投影与正北方向的夹角角速度, 单位为度/秒, 默认发送频率为 10Hz
gyroY	int32	Required	
gyroZ	int32	Required	
timeStamp	int64	Required	此次消息在车机端发出时的时间戳, 单位 ms

12.4. MSG_CMD_CAR_ACCELERATION

MSG_CMD_CAR_ACCELERATION 消息字段定义:

Field Name	Type	Optional/ Required	Description
accX	double	Required	水平方向加速度, 右为正, 单位 m/s^2
accY	double	Required	水平垂直方向加速度, 前为正, 单位 m/s^2
accZ	double	Required	空间垂直方向加速度, 天空方向为正, 单位 m/s^2

timeStamp	int64	Required	此次消息在车机端发出时的时间戳，单位 ms
-----------	-------	----------	-----------------------

12.5. MSG_CMD_CAR_OIL

MSG_CMD_CAR_OIL 消息字段定义：

Field Name	Type	Optional/ Required	Description
level	int32	Required	当前油量百分百（0-100）
range	int32	Required	当前油量能够行驶的距离（单位 km）
lowFuleWarning	bool	Required	是否油量过低

12.6. MSG_CMD_CAR_DATA_GEAR

MSG_CMD_CAR_DATA_GEAR 消息字段定义：

Field Name	Type	Optional/ Required	Description
gear	Int32	Required	当前挡位状态 enum{ GEAR_NEUTRAL = 1; GEAR_PARK = 2; GEAR_DRIVE = 3; GEAR_LOW = 4; GEAR_REVERSE = 5; } 其中， GEAR_NEUTRAL：空档； GEAR_PARK：驻车档； GEAR_DRIVE：动力档； GEAR_LOW：低速档； GEAR_REVERSE：倒车档；

12.7. MSG_CMD_CAR_DATA_SUBSCRIBE

MSG_CMD_CAR_DATA_SUBSCRIBE 消息字段定义：

手机查询车机支持订阅的所有数据类型，消息内容为空。

12.8. MSG_CMD_CAR_DATA_SUBSCRIBE_DONE

MSG_CMD_CAR_DATA_SUBSCRIBE_DONE 消息字段定义:

Field Name	Type	Optional/ Required	Description
cnt	int32	Required	车机支持订阅的数据模块个数， vehicleInfo 字段将为数组，大小为 cnt 个
vehicleInfo[]	message CarLifeVehicleInfo	Required	message CarLifeVehicleInfo 指车身信息，具体见下表

MSG_CMD_CAR_DATA_SUBSCRIBE_DONE 消息中 CarLifeVehicleInfo 字段定义:

Field Name	Type	Optional/ Required	Description
moduleID	int32	Required	车身信息模块 ID <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> 其中， CAR_DATA_GPS: 车身 GPS 模块 ID; CAR_DATA_VELOCITY: 车身速度模块; CAR_DATA_GYROSCOPE: 车身陀螺仪模块; CAR_DATA_ACCELERATION: 车身加速度模块; CAR_DATA_GEAR: 车身档位模块; CAR_DATA_OIL: 车身油量模块;
supportFlag	Int32	Required	0: 表示车机不支持该模块的数据报告 如果订阅的是 GPS 数据，改字段描述的是 GPS 数据的格式: 目前支持两种格式: 1: 支持 GPS 数据格式 WGS-84, 为不加密格式 2: 支持 GPS 数据格式为 GCJ-02,

			为加密格式 车机根据自己支持的格式来上报 GPS 数据。
frequency	Int32	Required	车身信息订阅频率（单位 Hz）； 该频率不能低于手机请求的频 率，如果低于手机请求频率，手 机会放弃使用该车机数据

12.9. MSG_CMD_CAR_DATA_SUBSCRIBE_START

MSG_CMD_CAR_DATA_SUBSCRIBE_START 消息字段定义：

Field Name	Type	Optional/ Required	Description
vehicleInfo	message CarLifeVehicleInfo	Required	手机请求订阅该类型数据，车机 可开始上报该类型数据。 message CarLifeVehicleInfo 指车身 信息，具体见下表

MSG_CMD_CAR_DATA_SUBSCRIBE_START 消息中 CarLifeVehicleInfo 字段端定义：

Field Name	Type	Optional/ Required	Description
moduleID	int32	Required	车身信息模块 ID 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; } 其中， CAR_DATA_GPS：车身 GPS 模块 ID； CAR_DATA_VELOCITY：车身速度 模块； CAR_DATA_GYROSCOPE：车身陀 螺仪模块； CAR_DATA_ACCELERATION：车身 加速度模块； CAR_DATA_GEAR：车身档位模块； CAR_DATA_OIL：车身油量模块；
supportFlag	Int32	Optional	(Reserved)

frequency	Int32	Required	车身信息订阅频率（单位 Hz）；
-----------	-------	----------	------------------

12.10. MSG_CMD_CAR_DATA_SUBSCRIBE_STOP

MSG_CMD_CAR_DATA_SUBSCRIBE_STOP 消息字段定义：

Field Name	Type	Optional/ Required	Description
vehicleInfo	message CarLifeVehicleInfo	Required	手机取消订阅该类型数据，车机应该停止上报该类型数据 message CarLifeVehicleInfo 指车身信息，具体见下表

MSG_CMD_CAR_DATA_SUBSCRIBE_STOP 中 CarLifeVehicleInfo 消息字段定义：

Field Name	Type	Optional/ Required	Description
moduleID	int32	Required	车身信息模块 ID 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; } 其中， CAR_DATA_GPS：车身 GPS 模块 ID； CAR_DATA_VELOCITY：车身速度模块； CAR_DATA_GYROSCOPE：车身陀螺仪模块； CAR_DATA_ACCELERATION：车身加速度模块； CAR_DATA_GEAR：车身档位模块； CAR_DATA_OIL：车身油量模块；
supportFlag	Int32	Optional	(Reserved)
frequency	Int32	Optional	(Reserved)

13. CarLife 可以给车机系统提供的数据

为了提升用户驾车过程中的体验，carlife 支持把以下信息传给车机端，由车机端可以把相关信息展示到车机主屏幕之外的其他显示区域。车机端可以自己选择是否订阅和处理这些消息，其中导航 TBT 信息遵循数据订阅机制，订阅消息请参考控制消息中 MSG_CMD_CARLIFE_DATA_SUBSCRIBE 消息定义。

13.1. 导航 TBT 信息

CarLife 在导航过程中，可以给车机端提供以下简易诱导信息，下一路口名字，距离下一个路口的距离，下一个路口的转向类型，刷新策略跟导航中简易诱导消息策略保持一致。传递消息时使用的相关通信协议详见下表。

该消息由手机发送到车机，具体消息中的字段定义如下表所示。

Field Name	Type	Optional/ Required	Description
action	int32	Required	TBT 信息更新类型（显示，更新，隐藏），具体请参考表格 13.1-2
naviTurn	int32	Required	下一个诱导的诱导类型，是个枚举值，具体请参考表 13.1-3
roadName	String	Required	下个诱导点的路名
totalDistance	int32	Required	距离下个诱导点的总距离，单位是米
remaindistance	int32	Required	距离下个诱导点的距离，单位是米
turnIconData	byte	Required	转向标图片的位图二进制数据

表 13.1-1 MSG_CMD_NAVI_NEXT_TURN_INFO 消息结构定义

Enum_Name	Value	Usage
Action_Invalid	0	无效值
Action_Show	1	显示
Action_Update	2	更新
Action_Hide	3	隐藏

表 13.1-2 TBT 消息更新类型

Enum_Name	Value	Usage
Turn_Kind_Front	0x00000001	直行
Turn_Kind_Right_Front	0x00000002	右前方转弯
Turn_Kind_Right	0x00000003	右转
Turn_Kind_Right_Back	0x00000004	右后方转弯
Turn_Kind_Back	0x00000005	调头
Turn_Kind_Left_Back	0x00000006	左后方转弯
Turn_Kind_Left	0x00000007	左转
Turn_Kind_Left_Front	0x00000008	左前方转弯

Turn_Kind_Ring	0x00000009	环岛
Turn_Kind_RingOut	0x0000000A	环岛出口
Turn_Kind_Left_Side	0x0000000B	普通/JCT/SAPA 二分歧靠左
Turn_Kind_Right_Side	0x0000000C	普通/JCT/SAPA 二分歧靠右
Turn_Kind_Left_Side_Main	0x0000000D	左侧走本线
Turn_Kind_Branch_Left_Main	0x0000000E	靠最左走本线
Turn_Kind_Right_Side_Main	0x0000000F	右侧走本线
Turn_Kind_Branch_Right_Main	0x00000010	靠最右走本线
Turn_Kind_Center_Main	0x00000011	中间走本线
Turn_Kind_Left_Side_IC	0x00000012	IC 二分歧左侧走 IC
Turn_Kind_Right_Side_IC	0x00000013	IC 二分歧右侧走 IC
Turn_Kind_Branch_Left	0x00000014	普通三分歧/JCT/SAPA 左侧
Turn_Kind_Branch_Right	0x00000015	普通三分歧/JCT/SAPA 右侧
Turn_Kind_Branch_Center	0x00000016	普通三分歧/JCT/SAPA 中央
Turn_Kind_Start	0x00000017	起始点
Turn_Kind_Dest	0x00000018	终点
Turn_Kind_VIA1	0x00000019	途经点 1
Turn_Kind_VIA2	0x0000001A	途经点 2
Turn_Kind_VIA3	0x0000001B	途经点 3
Turn_Kind_VIA4	0x0000001C	途经点 4
Turn_Kind_InFerry	0x0000001D	进渡口
Turn_Kind_OutFerry	0x0000001E	出渡口
Turn_Kind_TollGate	0x0000001F	收费站
Turn_Kind_Left_Side_Straight_IC	0x00000020	IC 二分歧左侧直行走 IC
Turn_Kind_Right_Side_Straight_IC	0x00000021	IC 二分歧右侧直行走 IC
Turn_Kind_Left_Side_Straight	0x00000022	普通/JCT/SAPA 二分歧左侧直行
Turn_Kind_Right_Side_Straight	0x00000023	普通/JCT/SAPA 二分歧右侧直行
Turn_Kind_Branch_Left_Straight	0x00000024	普通/JCT/SAPA 三分歧左侧直行
Turn_Kind_Branch_Center_Straight	0x00000025	普通/JCT/SAPA 三分歧中央直行
Turn_Kind_Branch_Right_Straight	0x00000026	普通/JCT/SAPA 三分歧右侧直行
Turn_Kind_Branch_Left_IC	0x00000027	IC 三分歧左侧走 IC
Turn_Kind_Branch_Center_IC	0x00000028	IC 三分歧中央走 IC
Turn_Kind_Branch_Right_IC	0x00000029	IC 三分歧右侧走 IC
Turn_Kind_Branch_Left_IC_Straight	0x0000002A	IC 三分歧左侧直行
Turn_Kind_Branch_Center_IC_Straight	0x0000002B	IC 三分歧中央直行
Turn_Kind_Branch_Right_IC_Straight	0x0000002C	IC 三分歧右侧直行
Turn_Kind_Straight_2Branch_Left_Base	0x0000002D	八方向靠左直行
Turn_Kind_Straight_2Branch_Right_Base	0x0000002E	八方向靠右直行
Turn_Kind_Straight_3Branch_Left_Base	0x0000002F	八方向靠最左侧直行
Turn_Kind_Straight_3Branch_Middle_Base	0x00000030	八方向沿中间直行
Turn_Kind_Straight_3Branch_Right_Base	0x00000031	八方向靠最右侧直行
Turn_Kind_Left_2Branch_Left_Base	0x00000032	八方向左转+随后靠左
Turn_Kind_Left_2Branch_Right_Base	0x00000033	八方向左转+随后靠右

Turn_Kind_Left_3Branch_Left_Base	0x00000034	八方向左转+随后靠最左
Turn_Kind_Left_3Branch_Middle_Base	0x00000035	八方向左转+随后沿中间
Turn_Kind_Left_3Branch_Right_Base	0x00000036	八方向左转+随后靠最右
Turn_Kind_Right_2Branch_Left_Base	0x00000037	八方向右转+随后靠左
Turn_Kind_Right_2Branch_Right_Base	0x00000038	八方向右转+随后靠右
Turn_Kind_Right_3Branch_Left_Base	0x00000039	八方向右转+随后靠最左
Turn_Kind_Right_3Branch_Middle_Base	0x0000003A	八方向右转+随后沿中间
Turn_Kind_Right_3Branch_Right_Base	0x0000003B	八方向右转+随后靠最右
Turn_Kind_Left_Front_2Branch_Left_Base	0x0000003C	八方向左前方靠左侧
Turn_Kind_Left_Front_2Branch_Right_Base	0x0000003D	八方向左前方靠右侧
Turn_Kind_Right_Front_2Branch_Left_Base	0x0000003E	八方向右前方靠左侧
Turn_Kind_Right_Front_2Branch_Right_Base	0x0000003F	八方向右前方靠右侧
Turn_Kind_Back_2Branch_Left_Base	0x00000040	八方向调头+随后靠左
Turn_Kind_Back_2Branch_Right_Base	0x00000041	八方向调头+随后靠右
Turn_Kind_Back_3Branch_Left_Base	0x00000042	八方向调头+随后靠最左
Turn_Kind_Back_3Branch_Middle_Base	0x00000043	八方向调头+随后沿中间
Turn_Kind_Back_3Branch_Right_Base	0x00000044	八方向调头+随后靠最右
Turn_Kind_Ring_Front	0x00000045	环岛向前
Turn_Kind_Ring_Left	0x00000046	环岛向左
Turn_Kind_Ring_LeftBack	0x00000047	环岛向左后
Turn_Kind_Ring_LeftFront	0x00000048	环岛向左前
Turn_Kind_Ring_Right	0x00000049	环岛向右
Turn_Kind_Ring_RightBack	0x0000004A	环岛向右后
Turn_Kind_Ring_RightFront	0x0000004B	环岛向右前
Turn_Kind_Ring_TurnBack	0x0000004C	环岛向后

表 13.1-3 诱导类型的枚举值

诱导类型对应的示意参考图请参考文档《CarLife 车机互联项目手机端发送给车机端导航 HUD 转向类型说明.docx》。

CarLife 在导航过程中或者电子狗过程中, CarLife 可以车机端提供以下辅助诱导信息, 包括: 辅助诱导标类型 (如测速摄像头), 距离对应诱导点距离以及诱导标图片数据。刷新策略跟导航及电子狗中辅助诱导消息策略保持一致。具体请参考下表消息结构定义。

Field Name	Type	Optional/ Required	Description
action	int32	Required	TBT 信息更新类型 (显示, 隐藏, 消失), 具体请参考表格 15-2
assistantType	int32	Required	辅助诱导消息类型, 是个枚举值, 具体请参考表 15-5
trafficSignType	int32	Required	交通诱导消息子类型, 是个枚举值, 当 assistantType 为 trafficSign 时有效, 保留字段, 暂未使用, 默认值为 0

totalDistance	int32	Required	距离下个辅助诱导点的总距离，单位是米
remainDistance	int32	Required	距离下个辅助诱导点的剩余距离，单位是米
cameraSpeed	int32	Required	测速摄像头限速值，单位米/秒，只有当 assistantType 为 Assistant_Type_SpeedCamera 时才有效

表 13.1-4 MSG_CMD_NAVI_AssitantGuide_INFO 消息结构定义

Enum_Name	Value	Usage
Assistant_Type_Invalid	0	无效值
Assistant_Type_trafficSign	1	摄像头之外的其他交通标志，保留类型，暂未使用
Assistant_Type_SpeedCamera	2	测速摄像头
Assistant_Type_TrafficLightCamera	3	交通信号灯摄像头
Assistant_Type_PeccanryCamera	4	违章摄像头
Assistant_Type_IntervalCamera	5	区间限速

表 13.1-5assistantType 类型枚举定义

13.2. 当前歌曲播放信息

CarLife 的音乐播放的过程中，可以给车机端提供当前正在播放歌曲的歌名，演唱者，歌曲长度。刷新策略是开始播放或者切换歌曲时刷新。具体请参考下表。

Field Name	Type	Optional/ Required	Description
source	string	Required	当前播放歌曲的源
song	string	Required	当前播放歌曲名字
artist	string	Required	当前播放歌曲演唱家
album	string	Required	当前播放歌曲专辑名
albumArt	byte	Required	当前播放歌曲封面图片的位图二进制数据
duration	int32	Required	当前播放歌曲长度，单位是秒
playlistNum	int32	Required	当前播放列表歌曲数量
songId	string	Required	当前播放的歌曲 ID(CarLife 内部使用)
mode	int32	Required	播放模式，是一个枚举，定义见表 13.2-2

表 13.2-1 MSG_CMD_MEDIA_INFO 结构定义

Enum_Name	Value	Usage
PLAYMODE_REPEATE_ONE	0	单曲循环
PLAYMODE_SHUFFLE	1	随机播放
PLAYMODE_REPEATE_ALL	2	循环播放

表 13.2-2 PlayMode 枚举定义

13.3. 当前歌曲播放进度信息

CarLife 的音乐播放的过程中，可以给车机端提供当前正在播放歌曲的进度信息。刷新策略为每秒一次。具体请参考下表。

Field Name	Type	Optional/ Required	Description
progressBar	Int32	Required	当前播放歌曲的进度信息（单位 ms）

表 13.3-1 MSG_CMD_MEDIA_PROGRESS_BAR 结构定义

14. CarLife 车机端退出机制

在联机状态下，点击 CarLife 首页的“CarLife”按钮进去的子页面，有“退出到车载系统”的按钮，用户点击此按钮，手机端会向车机端发 MSG_CMD_GO_TO_DESKTOP 消息，车机端可根据自身需求选择以下两种处理方式：

- 完全退出车机端
- 切换到车机系统主页

15. 更新机制

15.1. 车机端

作为一款高度符合车规级规范的车机端方案，CarLife 的车机端特性会在很长一段时间内保持稳定，一般情况下无需更新，所有的功能更新都在手机侧。但为了系统的全面性，我们还是允许了符合车厂习惯的车机端相关更新的策略，车厂可以根据自身系统的情况选择以下任何一种车机端更新机制。

- 4S 店面更新
- OTA 更新

当 CarLife 车机端有重大更新时，我们会及时把信息分享给合作车厂。车厂可以根据自身情况决定是否更新车机端以支持最新的特性。

车厂在做车机端更新之前，必须经过 CarLife 团队的验收流程。

15.2. 手机端

CarLife 手机端会根据内部版本计划和节奏，定期的在各大 Android 应用市场和 iPhone appstore 进行版本更新。这里的流程与其它手机端的 app 更新机制无异。

15.3. 版本兼容机制

15.3.1. 车机端

CarLife 车机端在设计上是一个非常轻量级的架构，几乎没有任何产品逻辑，只要连接、音视频、反控等基础功能接入完成，就可保证 CarLife 的基础功能可用。后续的更新只会新增特性而不会修改原有的特性，因此，只要车厂遵循 CarLife 的协议规范和接入指南，车机端的新旧版本之间是兼容的。

15.3.2. 手机端

CarLife 手机端的任何更新，不会影响到与车机端版本的兼容。

15.3.3. 数据协议

车机端和移动设备端使用的数据协议存在兼容性问题，各自使用的数据协议有相应的版本号，由两位表示，第一位为主版本号，第二位为次版本号。目前在 CarLife 项目中车机端和移动设备端协议版本号被初始化为 1.0。车机端和移动设备端数据协议兼容规则如下：

- 当车机端和移动设备端数据协议主版本号相同时，则车机端和移动设备端数据协议兼容；
- 如果旧数据协议和新的数据协议不兼容，比如修改了原有数据协议的 pb 结构等，车机端和移动设备端新版本数据协议主版本号都应该增加 1，次版本号都置为 0；
- 一般来讲，新版本协议相对于旧版本协议只会增加协议而不会去修改原有的协议，以此来保证新旧协议的兼容性。新协议增加了移动设备端传递给车机端新的消息（该消息不影响正常的功能，仅传递数据，车机端可以处理该数据也可以不处理）等，车机端和移动设备端新版本数据协议次版本号都应该增加 1，主版本号都不发生变化。

16. 安全认证

16.1. 手机端 CarLife 对车机端 CarLife 安全认证

手机端 CarLife 通过安全认证流程，来确认正在进行连接的车机，符合 CarLife 规范要求。同时，手机端 CarLife 通过认证过程来辨识连接车机的具体类型，从而提供定制性的服务和内容。

CarLife 的安全认证为选配功能，车机厂商可以根据情况选择性支持。

16.1.1. 认证消息定义

手机端 CarLife 对车机端 CarLife 的认证通过 MSG_CMD_STATISTIC_INFO 消息来实现。具体的消息定义请参考 10.2.2 章节。

其中，MSG_CMD_MD_AUTHEN_RESULT 消息的字段定义如表 7-4 所示：

表 17.2-1: MSG_CMD_MD_AUTHEN_RESULT 消息字段定义

Field Name	Type	Optional/ Required	Description
result	bool	Required	手机端 CarLife 对车机端 CarLife 的验证结果

16.1.2. 认证流程

手机端 CarLife 对车机端 CarLife 的认证流程如图 17.2-2 所示：

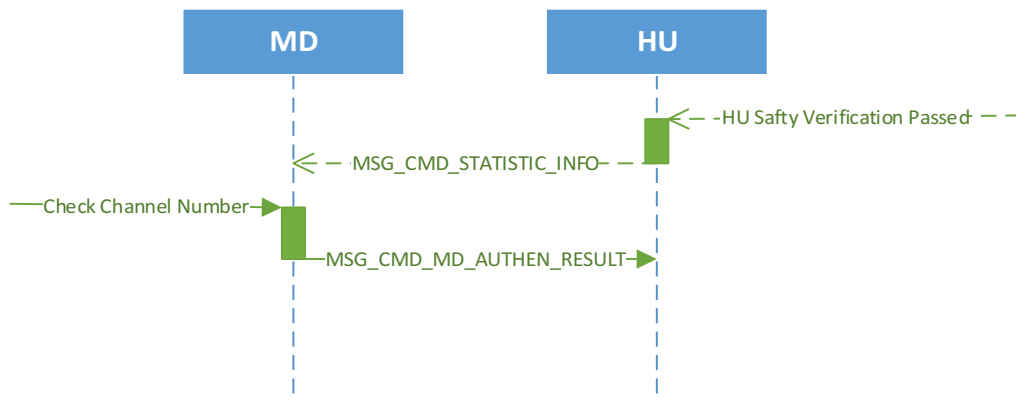


图 17.2-2: 手机端 CarLife 对车机端 CarLife 认证过程

1. 在车机端 CarLife 对手机端 CarLife 的认证通过之后，车机端将统计信息 MSG_CMD_STATISTIC_INFRO 信息发送到手机端 CarLife；
2. 手机端 CarLife 接收到验证统计信息后，通过渠道号来确认该车机是否为合法车机；
3. 如果车机为合法车机，则手机端 CarLife 对车机端 CarLife 认证结束。手机端通过 MSG_CMD_MD_AUTHEN_RESULT 消息将验证结果发送到车机端，并启动车厂相关的定制服务；
4. 如果车机为非法车机，则手机端通过 MSG_CMD_MD_AUTHEN_RESULT 消息将验证结果发送到车机端，并主动断开连接；

17. 性能指标

性能指标主要用于反映车机运行 CarLife 时整体性能及与用户相关的体验指标，具体如下：

表 18-1:

序号	指标项	说明
#1	CPU	车机端 CPU 指标用于保障车机端正确使用 CPU，主要指车机端进程占用系统整体 CPU 百分比，如使用 top 命令进行采集 CPU 占用，最大使用场景 CPU 平均建议小于 20%；
#2	内存	车机端内存指标用于保障车机端正确合理申请内存，反映各场景下(包括但不限于导航、音乐、音乐播放+导航等)及各页面操作后内存占用情况，主要指车机端进程占用系统内存大小，建议最大使用场景内存不能超过 50M。
#3	视频帧率	视频帧率用于衡量手机、车机互联后视频转输帧率及流畅度；车机端需要保证其有足够的解码能力，视频解码帧率至少需支持 30 帧/s 以上，视频无明显卡顿延迟失帧问题。
#4	视频延时	视频延迟用于衡量手机、车机互联后两端视频同步性是否符合要求，从而达到良好的用户使用体验，建议视频延迟控制在 200 ms 以下。
#5	MIC 功能	语音识别率用于衡量车机 MIC 是否符合 CarLife 使用要求，使用录音软件在车机端进行录音并对其语料进行分析无车厂自身噪音（杂音不影响语音）或截断现象，详见 8.7 VR 功能说明；
#6	连接时长	连接时长用于衡量手机与车机建立连接的耗时是否符合要求，连接过程中不会引起用户焦虑，建议连接时长控制在 4s 以内。

18. 统计信息声明

为了提供更好的服务质量，CarLife 需要通过车机与手机的协议，将车机中的相关信息传输到手机端，由手机端进行上报统计。协议中与此相关的消息定义有两条：（1）

MSG_CMD_HU_INFO，车机系统信息；（2）MSG_CMD_STATISTIC_INFO，渠道及运行统计信息。下文将会详细介绍这两条消息中用到的字段的含义及用途。

18.1. 车机系统统计信息声明（MSG_CMD_HU_INFO）

车机系统统计信息，也即是上文所提到的 MSG_CMD_HU_INFO，此消息包含了车机系统的基本硬件及软件信息，获取这些信息用于跟踪和定位问题，帮助我们更快地找到问题根源，从而更快解决问题。假如出现了线上 BUG，我们根据上报的车机系统信息，快速定位出问题的车机，还原出 BUG 的复现环境（硬件及软件环境），从而可以更快地分析定位问题所在，迅速修复 BUG。

此消息中包含了以下字段，分别对应各种系统信息：

字段名	类型	可选/必选	说明
os	String	可选	车机操作系统类型，包括 QNX、Linux、WinCE 和 Android
manufacturer	String	可选	车机制造商的名称，例如 Hyundai
model	String	可选	车机型号，例如 HD_MODEL_112
release	String	可选	车机系统发布版本名称，例如 HDCS1.0

表 19.1-1

以上字段均是可选字段，提供更多的信息有助于问题的跟踪和解决，我们建议合作方尽量提供更多信息。在连接成功后，HU 端主动将以上信息发送到 MD 端，由 MD 端保存，在程序出现异常时会将此信息上报供分析使用。

18.2. 渠道及运行统计信息声明（MSG_CMD_STATISTIC_INFO）

渠道及运行统计信息，也即是上文所提及的 MSG_CMD_STATISTIC_INFO，此消息包含了车机的渠道信息及运行信息，获取这些信息用于上报车机端的运行情况与运行日志，帮助我们提升产品的稳定性，增强产品的适配性，优化产品的体验。

此消息中包含以下字段，它们分别代表不同的含义：

字段名	类型	必选/可选	说明
cuid	String	必选	车机的唯一标识码，用于唯一标识每一台车机
versionName	String	必选	车机端的版本名，例如 1.0.0，1.1.0
versionCode	Integer	必选	车机端的版本号，例如 1，2
channel	String	必选	车机端的渠道号，由百度统一分配，用于唯一标识一款车机产品
connectCount	Integer	必选	车机端与手机端连接次数，由车机端动态保存，连接成功后，该参数需要清零，用于统计连接的成功率
connectSuccessCount	Integer	必选	车机端与手机端连接成功次数，一般设置为 1，用于统计连接的成功率

connectTime	Integer	必选	车机端与手机端连接耗时,用于统计平均连接耗时
crashLog	String	可选	车机端上次异常退出的关键日志,用于跟踪问题,修复问题

表 19.2-1

在车机端与手机端连接成功后,车机端必须在 2 分钟内将此消息发送至手机端(具体时序参考时序图),否则手机端将会主动断开与车机端的连接。手机端在收到此消息后会将上述信息上传到服务器中,用于统计车机端的使用情况及异常日志。

19. CarLife 团队对于车厂提供的支持

19.1. 主要测试工具

19.1.1. CarLife 协议工具

主要用于检测、输出手机端、车机端建立连接时信息交互、手机端退后台、硬按键、音频及其它实时交互信息,方便开发过程中问题定位及消息确认。

车机与 Android 端手机互联时协议消息分析操作步骤:

1. 使用百度分配的账号密码登录 <http://180.76.233.125:8080/login.jsp> 网站,添加测试 Android 手机的 IMEI 号
2. 在测试手机安装百度提供的自测专用安装包(已开启手机车机协议消息上报功能和模拟导航功能)
3. 保持手机端网络连接,启动手机端 CarLife 程序,进行功能测试;在 Web 网站选中测试设备,协议消息会实时展示到 Web 界面

车机与 iOS 端手机互联时协议消息分析操作步骤:

1. PC 端(MAC 或者 Windows)安装 RemoteLogger 工具,保证 PC 端和测试手机在同一 WIFI 网络(该局域网支持广播)
2. 测试手机安装百度提供的自测专用安装包(已开启手机车机协议消息上报和模拟导航功能)
3. 测试手机和 RemoteLogger 工具连接成功后,将测试手机与车机端连接,进行功能测试
4. PC 端 RemoteLogger 工具会实时展示协议消息日志信息

20. CarLife 认证

为了更好的保障 CarLife 车机手机互联产品的质量及用户体验,百度方制定了此认证流程。如验收不通过,则需要合作方进行修复并重新发起验收直至验收通过才可展开后续的量产工作。

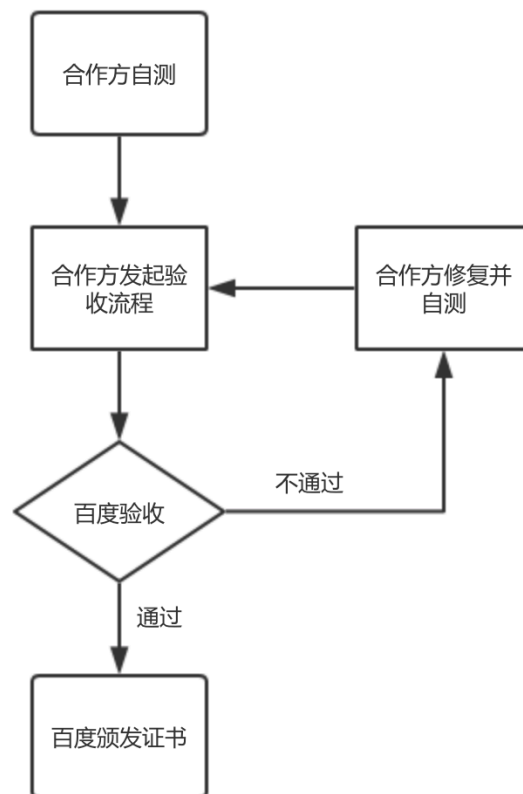


图 21-1: CarLife 认证流程

20.1 合作方自测并发起验收流程

版本质量主要由合作方保证，因此合作方完成开发后需进行系统测试工作，相关问题可通过《CarLife 车机手机互联问题.xls》进行跟踪，问题模块请参考《CarLife 车机手机互联问题模板.xls》填写。

合作方进行充分自测后：

1.可以进入官网 (<http://carlife.baidu.com/>) -> 登录百度账号 -> 点击“我是车厂/Tier1” -> 进入“我的项目”，选择所要发起认证的项目填写认证所需资料即可，需要提供正确的品牌车系车型中英文名称、寄送车机快递单号及项目自测报告。（前提是在 CarLife 官网 -> 我是车厂/Tier1 中有立项信息）

2.或者向百度方提出验收申请(发邮件到邮箱：carlife@baidu.com)。

以下为认证所需的材料详述：

1、CarLife 车机端软、硬件环境

a) 安装有最新车机端 CarLife 的车机系统及使用 CarLife 过程中需要的相关附属配件(如：麦克风、电源、喇叭)；

注：若车机系统或 CarLife 版本有更新请及时通知 CarLife 团队更新。

2、相关测试报告及文档

具体如下表 21.1-1：

表 21.1-1

编号	名称	形式	介质
----	----	----	----

1	启动 CarLife 说明（车机端）	文档	电子/纸质
2	系统参数配置说明	文档	电子/纸质
3	系统功能测试报告	文档	电子/纸质
4	系统性能测试报告	xls	电子/纸质
5	稳定性测试报告	xls	电子/纸质
6	兼容性测试报告	xls	电子/纸质
7	问题记录表	xls	电子/纸质

- 1) 启动 CarLife 说明（车机端）——简要说明用户如何通过合作方的车机端与手机端 CarLife 进行互联；
- 2) 系统参数配置需要说明——使用的渠道号、协议版本、统计项等数据说明；
- 3) 系统功能测试报告——基本功能测试通过，具体参考《CarLife 车机端基本功能 Case.xls》；
- 4) 系统性能测试报告——具体参考《CarLife 车机手机互联方案接入详细指南.docx》表 19 性能指标项；
- 5) 稳定性测试报告——建议在导航&音乐的状态下连续运行 **4 小时*3 次**（车机端无断连接、崩溃现象, 建议路测）
- 6) 兼容性测试报告——包括但不限于手机操作系统、版本、品牌、屏幕分辨率、手机端 CarLife 版本等兼容性测试。
- 7) 问题记录表——合作方和百度共同维护此文档，在开发、测试、验收、回归过程中都以此文档为主，具体模板请参考《CarLife 车机手机互联问题模板.xmls》；

20.2 百度验收

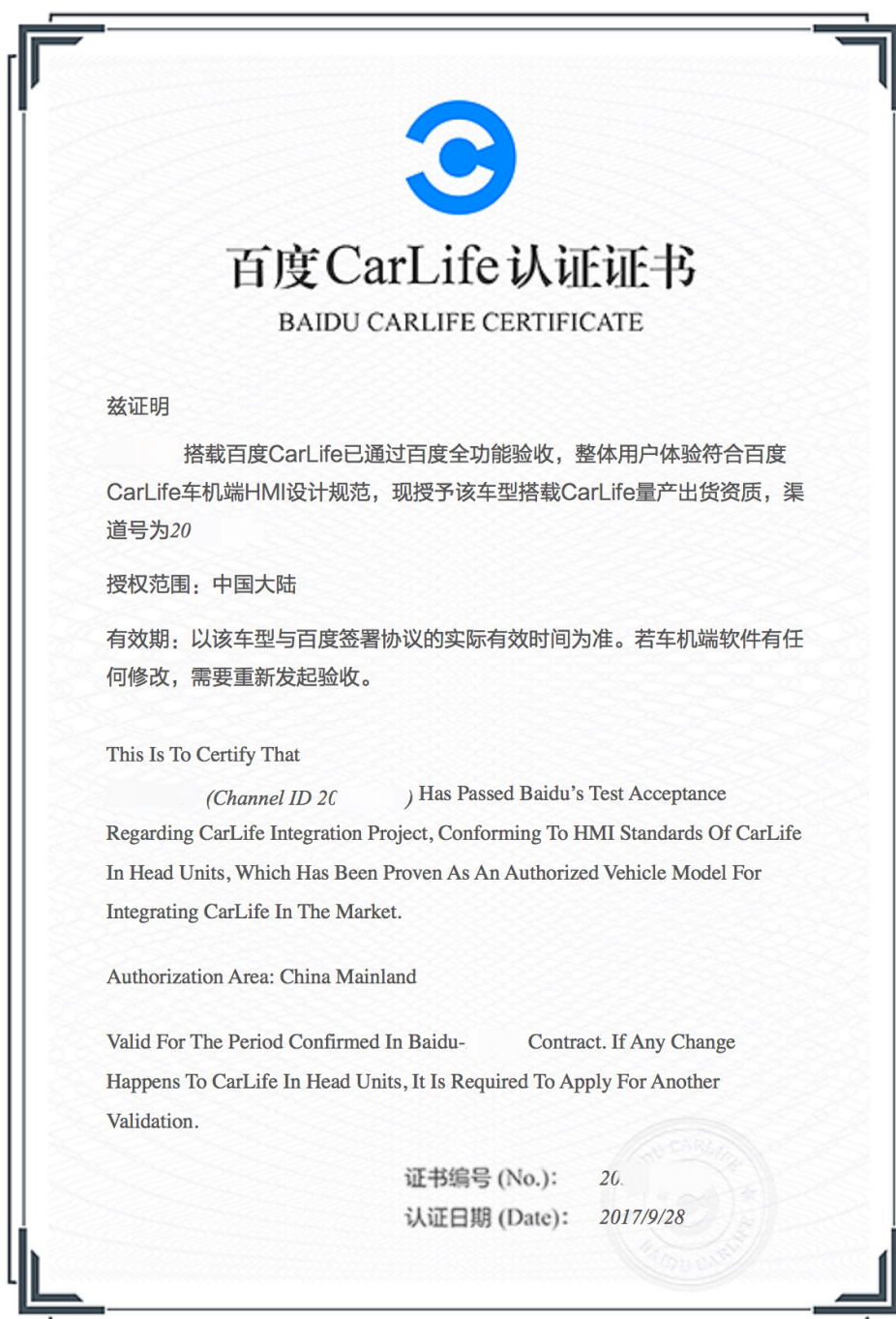
合作方发起验收流程后，百度根据其提供的文档和报告进行一系列的全功能测试验收，全功能测试结束后百度将给出对应的测试报告及问题列表。百度根据《车机端 CarLife 程序验收通过标准》来判定验收是否通过。

验收通过标准：1.无 P0、P1 级别 Bug，P2、P3 级别 Bug 数≤3 个（Bug 级别定义见《CarLife 车机端 Bug 级别定义》）2.车机端 CarLife 必须同时支持 Android 和 iOS 手机(若仅支持 Android 手机则验收无法通过)；3.Android 手机的 USB 有线连接方案必须支持 AOA(若仅支持 ADB 则验收无法通过)。详见《CarLife 车机端程序验收通过标准》文档。

验收阶段，百度侧进行以下三项工作，耗时 1~2 周：

1. 验收合作方提供的报告，包括功能测试、性能测试、兼容性测试、稳定性测试报告等；
2. 对车机进行测试，包括视频、音频、连接、语音、反控等功能；
3. 记录验收过程中发现的问题，需将其录入到问题记录表并将结果同步给合作方共同跟进解决；关于测试过程中问题的解决机制：
 - a) 若车机端导致用户不能正常使用的问题，则必须解决；
 - b) 针对偶现率极低且不影响用户正常使用问题，建议降低优化级后续跟进处理；
 - c) 手机端问题不作为车厂上车评测标准，其问题随手机端版本进行更新修复。

百度验收通过后将会为该项目颁发量产证书，证书样式如下：



21. 车机端 HMI 设计规范简述

为帮助各合作方更好地理解 CarLife 集成在车机端后的最终呈现效果,根据现有合作资源与合作模式,简要整理以下相关视觉及交互内容以供参考。具体请参考另一份详细文档《CarLife 车机端接入 HMI 规范》。

21.1. 车机端 CarLife Icon 及名字规范

车机端需要在相关菜单页面设有 CarLife 入口，为此我们向合作方提供 CarLife Icon（500x500）标准文件，由合作方自行基于车机端标准应用 icon 进行尺寸的适配。另外，针对希望添加 CarLife 地图快捷入口的合作方，我们将提供地图快捷入口 Icon。

图 22.1-1 CarLife Icon 标准文件效果图



图 22.1-2 CarLife 地图快捷入口 Icon 标准文件效果图



图 22.1-3 车机端菜单界面 CarLife Icon 范例图



中英文名称将随车机端系统语言进行改变，名称规范请遵循如下格式：

- 中文名称，百度 CarLife
- 英文名称，Baidu CarLife

**请注意 Baidu 和 CarLife 的大小写区分。*

21.2. 车机端连接动画规范

用户在使用 CarLife 过程中，涉及到诸多与连接相关的非视频流页面，需要由车机端做出相应的提示。接下来的内容，将总结介绍在不同连接场景车机端需要如何提示用户。合作方可选择使用 CarLife 官方设计页面或自行针对场景设计。

**以下图片为交互示意图仅供参考，并非实际效果图*

21.2.1. 连接前

用户在未连接手机的情况下，直接点击车机端菜单中的 CarLife Icon，车机端将提示用户连接手机和车载系统。

图 22. 2-1 未连接手机时，车机端示意图



21.2.2. 手机未安装 CarLife

用户连接手机后，点击车机端 CarLife icon，若车机端检测到手机中没有安装 CarLife，将提示用户扫描二维码下载安装，并点击车机端操作重连：

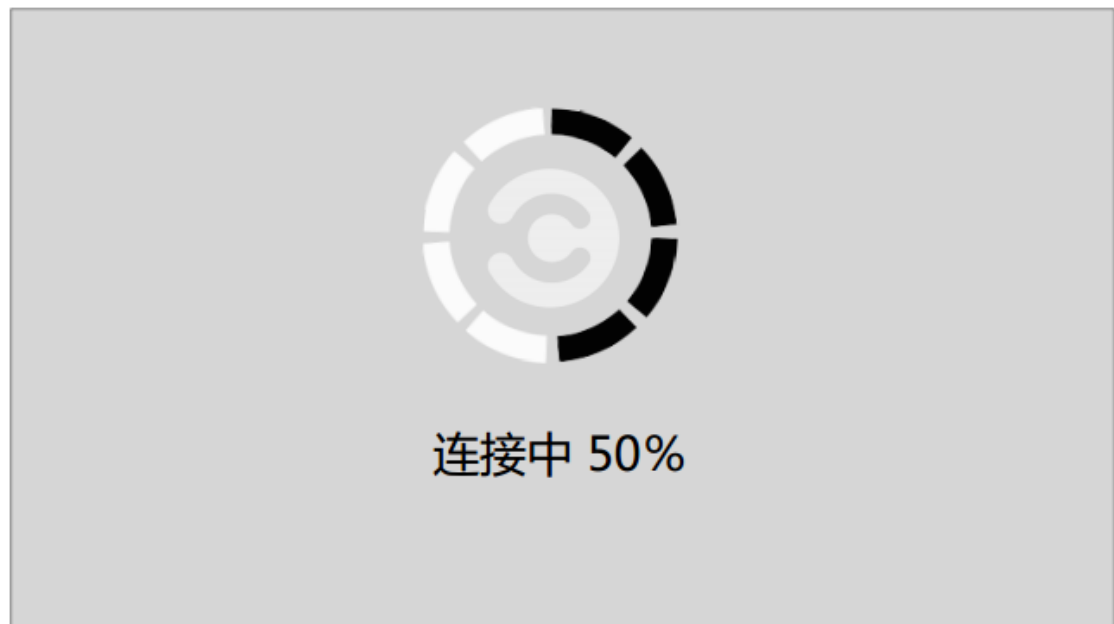
图 22.2-2 车机端检测到手机未安装 CarLife 示意图



21.2.3. 连接中

用户连接手机后，车机将进入连接载入状态，此时车机端将显示载入动画，如下图：

图 22. 2-3 连接中车机端提示动画示意图



21.2.4. 连接成功

当连接成功时，车机端将显示连接成功动画，随后将进入 CarLife 启动页和主页。

图 22.2-4 车机端连接成功动画



21.2.5. 连接失败/断开

当用户在连接载入界面中断连接，或者车机端检测到连接失败时，用户可操作重新连接或查看提示帮助。

图 22.2-5 车机端连接失败/断开示意图



21.2.6. 连接帮助

当用户断开连接或连接失败时，都可通过连接失败图示页右上角的图标进入车机端连接帮助。

图 22. 2-6 连接失败车机端连接帮助页示意图



21.2.7. 连接中 CarLife 切到后台/手机锁屏

连接中若用户将 CarLife APP 切到后台或将手机锁屏，车机端将提示用户此时 CarLife

无法使用。继续使用 CarLife，则需解锁手机，并在提示页点击‘启动 CarLife’按钮，使 CarLife 重回前台。

图 22.2-7 车机端提示 CarLife 处于后台/手机锁屏

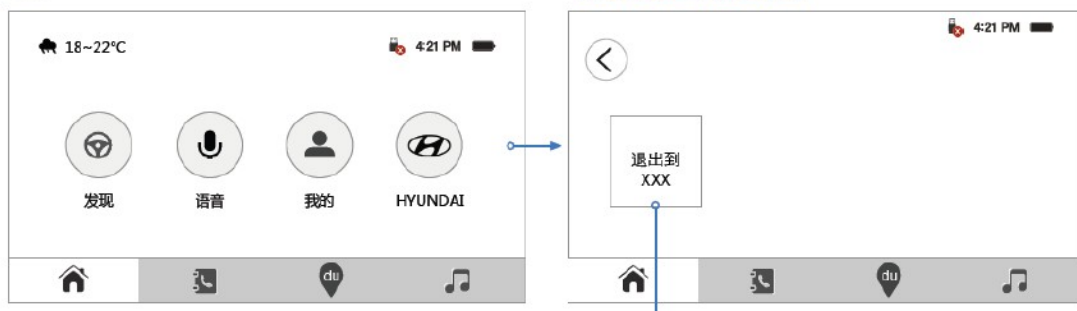


21.2.8. 退出/返回 CarLife

连接手机后，点击进入首页的 CarLife 按钮/汽车品牌按钮，可实现 CarLife 与原声车机系统的切换。根据合作方的需求，有如下两个选择：

- 返回车机原生系统
- 直接退出 CarLife

图 22.2-8CarLife 按钮/汽车品牌按钮示意图



21.2.9. 蓝牙未连接

若手机端与车机端蓝牙未连接，此时无论是否已适配 CarLife 蓝牙电话功能，车机端应显示 CarLife 通话中提示页。

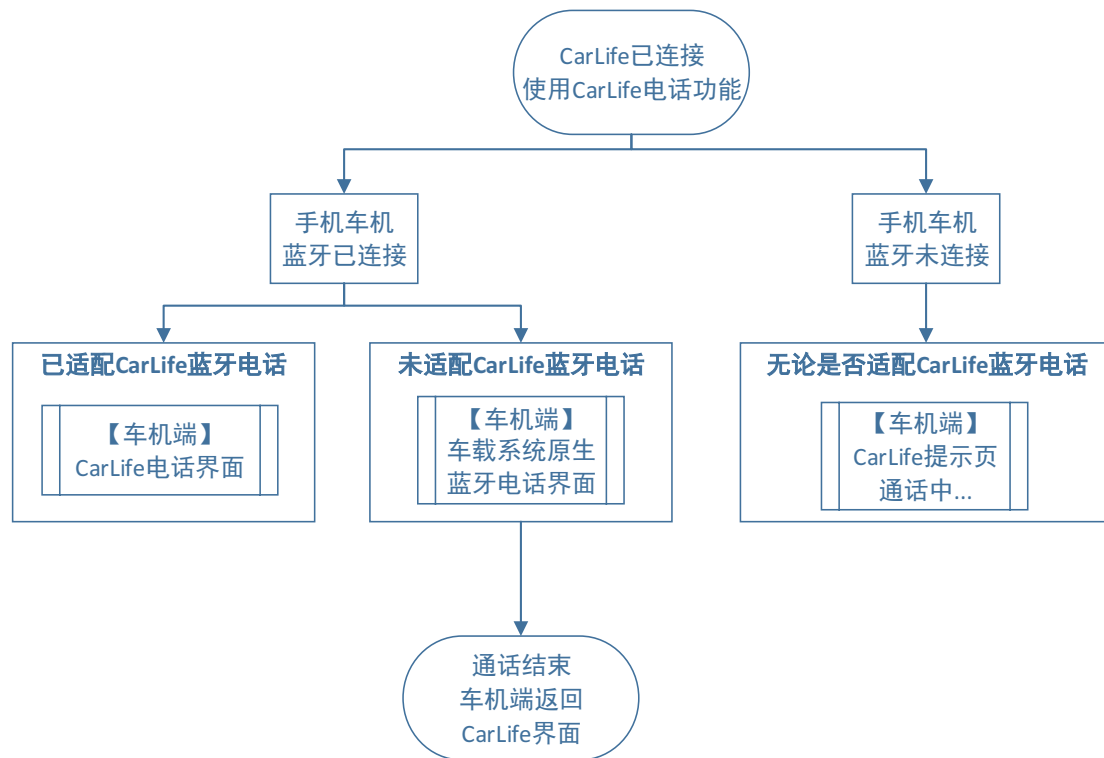
图 22. 2-9 车机端 CarLife 通话中示意图



21.3. 电话功能

手机端 CarLife 与车机端成功连接后，使用 CarLife 电话功能涉及到一系列与车机端的交互。实现 CarLife Internal UI 接听拨打电话功能，需保证手机与车机蓝牙已连接，车机端同时已完成 CarLife 蓝牙电话的适配工作。

图 22. 3-1 蓝牙电话车机交互实现概况



21.3.1. 蓝牙已连接

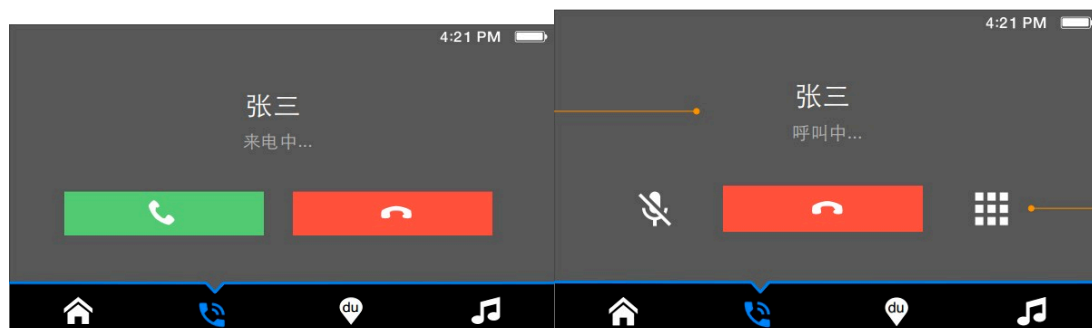
- 未适配 CarLife 蓝牙电话

当手机端与车机端蓝牙已连接，此时若车机端未适配 CarLife 蓝牙电话功能，车机端应显示车载系统原生蓝牙通话界面。

- 已适配 CarLife 蓝牙电话

当手机端与车机端蓝牙已连接，同时车机端也已适配 CarLife 蓝牙电话功能，车机端应显示 CarLife Internal UI 电话界面。

图 22.3-2CarLife Internal UI 来电&呼叫示意图



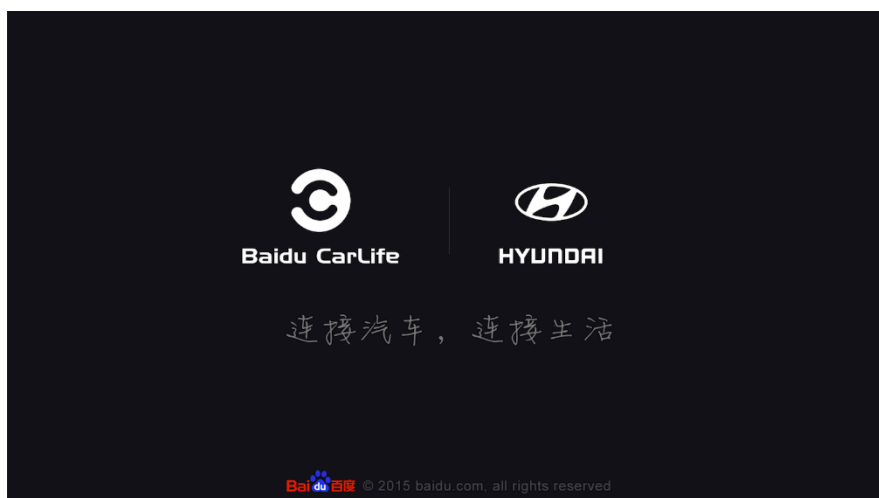
21.4. 车机端合作方品牌定制规范

针对手机端 APP 的 UI 界面设计，CarLife 为各合作车厂提供了两处品牌定制选项：启动页与 APP 主页品牌按钮。需合作方向 CarLife 提供品牌 LOGO 的源文件（AI 或 PSD）以便定制工作，具体定制效果图都将与合作方最终确认。

21.4.1. 车机端启动页

当车机端的 CarLife 被点击，连接初始化之前车机将显示 CarLife 车机端启动页。此时将共同展示 CarLife LOGO（左）与汽车品牌 LOGO（右），默认使用车厂品牌 LOGO 的白色版本，具体效果如下图所示：

图 22.4-1 现代车机端启动页效果图



21.4.2. 合作方品牌按钮

连接后，当手机检测到指定汽车品牌，首页第四个圆形按钮将自动变为车厂品牌按钮，我们会默认使用车厂品牌 LOGO 的白色版本以及品牌英文名称，如下图所示：

图 22.4-2 CarLife 首页戴姆勒汽车效果图



21.5. 默认激活

车机端中需要设有 CarLife 相关的设置项，当车机端有其他同类互联应用共存时，需直接激活并默认启动 CarLife。同时用户可在车机端设置中更改默认启动项。