



WeLink 车机端 APL 使用指南

Version 2.0.7

变更履历

版次号	修改说明	修改人	修改日期
1.0.0	初版作成	吕显洋	2017/05/23
1.0.1	追加“WL_MSG_TYPE_CAR_SER_NUM”消息; ·APL 提供物存放位置说明;	高奥	2017/05/25
1.0.3	追加录音相关接口	吕显洋	2017/06/20
1.0.6	库名及部分示例代码修正	闫海滨	2017/12/05
1.0.8	针对不同方案添加说明 (红字标注内容)	吕显洋	2017/12/19
1.0.9	接口变更, 追加无线相关接口	吕显洋	2018/01/10
1.0.10	追加加密相关接口	李楠	2018/01/25
1.0.13	时序图说明追加	李楠	2018/03/22
1.0.14	无线设备节点追加 Mac 地址信息	李楠	2018/03/23
1.0.15	设备类型中追加 AOA 设备 消减部分界面以及消息	李楠	2018/03/28
1.0.16	追加 APL 使用步骤说明 追加 Media 暂停/恢复消息	李楠	2018/04/16
2.0.0	追加各功能详细说明	WeLink 团队	2018/05/18
2.0.1	接口整理	吕显洋	2018/06/15
2.0.2	追加接口需要根据项目定制的说明 (12 章)	吕显洋	2018/07/03
2.0.3	追加 WeLink 公众号二维码画面和权限验证中画面 (9 章、12.4.5、12.4.6)	金瑀煊	2018/07/05
2.0.4	1、图像数据回调函数追加(5.1、12.2.5、12.3.1); 2、H.264 图像数据结构体变更(12.3.3);	吕显洋	2018/07/06
2.0.5	1、调整页面枚举的顺序(12.4.5) 2、追加导航条 icon 字段详细说明(12.3.7) 3、更新音频功能、触控功能介绍(6 章、7.2)	金瑀煊	2018/07/09
2.0.6	1、时序图修改(4 章)	黄庆余	2018/07/13
2.0.7	1、结构体名称错误修改 (12.3.7) 2、物理按键枚举修改 (12.4.7)	吕显洋	2018/07/31

	3、文件说明修改 (13.1)		
--	-----------------	--	--

WeLink

目录

1. 概述	8
1.1 前言	8
1.2 术语与缩略语	8
1.3 参考文档	8
2. 车机端 WeLink 架构说明	9
2.1 WeLink 方案整体框架	9
2.1.1 传输媒介	9
2.1.2 数据流	9
2.2 WeLink 方案车机端架构	10
2.2.1 WeLink APL 功能	11
2.2.2 车机端 APP 需要对应的功能	11
3. 车机平台软硬件需求	13
4. 连接方式	14
4.1 AOA 连接	14
4.1.1 AOA 简介	14
4.1.2 AOA 配件切换说明	14
4.1.3 AOA 通道的建立	16
4.1.4 车机端的开发工作	17
4.2 EAP 连接	18
4.2.1 EAP 简介	18
4.2.2 车机端的开发工作	18
4.2.3 EAP 通道的建立	18
4.3 WIFI 连接	20
4.3.1 WIFI 机能简介	20
4.3.2 WIFI 设备检测	20
4.3.3 车机端的开发工作	22
4.4 ADB 连接	22
4.4.1 ADB 简介	22
4.4.2 车机端的开发工作	23
4.5 WeLink 连接时序	23

5.	视频引擎	25
5.1	视频流	25
5.2	屏幕显示区域的设定	25
6.	音频引擎	26
6.1	音频播放	26
6.2	MIC 录音	26
7.	触控 (Touch)	27
7.1	车机端的开发工作	27
7.2	触控坐标	27
8.	消息通信 (Message)	27
8.1	APL 反馈给车机端 APP 的消息	27
8.2	APP 通知 WeLink APL 的消息	28
8.3	消息交互时序图	29
9.	页面简述 (Page)	30
9.1	车机端需要描画的页面	30
9.2	页面上的 button	36
10.	蓝牙	38
11.	硬按键	38
12.	APL 接口说明	39
12.1	API	40
12.1.1	wl_apl_set_config	40
12.1.2	wl_apl_init	41
12.1.3	wl_apl_deinit	41
12.1.4	wl_apl_point_down	41
12.1.5	wl_apl_point_up	41
12.1.6	wl_apl_move	42
12.1.7	wl_apl_send_button	42
12.1.8	wl_apl_send_msg	42
12.1.9	wl_apl_get_version	43
12.1.10	wl_apl_send_record_data	43
12.1.11	wl_apl_send_audio_data	44
12.1.12	wl_apl_send_point	44

12.1.13	wl_apl_send_point_sync	45
12.1.14	wl_apl_send_hard_key	45
12.1.15	wl_apl_device_chg.....	45
12.1.16	wl_apl_get_wlan_list	46
12.1.17	wl_apl_send_hu_data.....	46
12.2	回调函数	47
12.2.1	update_page.....	47
12.2.2	recv_msg.....	48
12.2.3	audio_data.....	50
12.2.4	hu_data.....	50
12.2.5	video_data	50
12.3	数据结构	51
12.3.1	APL 回调函数	51
12.3.2	RGB 图像信息	51
12.3.3	H.264 数据信息.....	52
12.3.4	图像数据信息.....	52
12.3.5	屏幕相关信息.....	52
12.3.6	APL 配置信息	52
12.3.7	导航条信息.....	53
12.3.8	无线设备节点.....	55
12.3.9	加密相关信息.....	55
12.3.10	USB 设备信息	55
12.4	枚举	55
12.4.1	H.264 图像类型.....	56
12.4.2	RGB 图像格式	56
12.4.3	图像数据格式定义.....	56
12.4.4	通信消息类型定义.....	57
12.4.5	页面类型定义.....	58
12.4.6	页面按键类型定义.....	59
12.4.7	物理按键.....	60
12.4.8	触控类型.....	60
12.4.9	设备类型.....	60

13.	APL 的具体使用	61
13.1	APL 文件说明	61
13.2	APL 使用步骤	61
13.2.1	配置并初始化 WeLink APL.....	61
13.2.2	手机设备接入与断开.....	62
13.2.3	处理 APL 返回的消息.....	62
13.2.4	向 APL 发送消息.....	62
13.2.5	页面更新处理.....	62
13.2.6	音频数据的播放.....	62
13.2.7	语音识别.....	63
13.2.8	回控的使用.....	63
13.2.9	其他	63
14.	FAQ	64

1. 概述

1.1 前言

本文档是趣驾 WeLink 车机互联方案的接入详细指南，重点针对 WeLink APL（下文简称为 APL）接入方案，对车机厂商或者供应商在车机中接入 WeLink 时所需的各方面工作，进行详细的描述。

趣驾 WeLink 是趣驾产品中的“手-车”互联解决方案，能够将 IOS、Android 手机的使用延展融合到车载环境。

1.2 术语与缩略语

术语/缩略语	描述
HU	车机：车载娱乐信息系统（Head Unit）。
MU	手机（Mobile Unit），也指文中的手机端。
UI	User Interface（用户界面）的简称。
SDK	Software Development Kit（软件开发工具包）。
JSON	(JavaScript Object Notation) 是一种轻量级的数据交换格式。
Mic	麦克风。
BlueTooth	蓝牙，本文中简写为 BT。
launcher	这里指 WeLink 手机端的应用程序。
VR	Voice Recognition 语音识别功能

1.3 参考文档

No.	文档名称
1	WeLink 车机端 EAP 说明文档（车机端_Linux）
2	WeLink 音频数据通信协议

2. 车机端 WeLink 架构说明

2.1 WeLink 方案整体框架

WeLink 方案整体框架如图 2-1 所示。

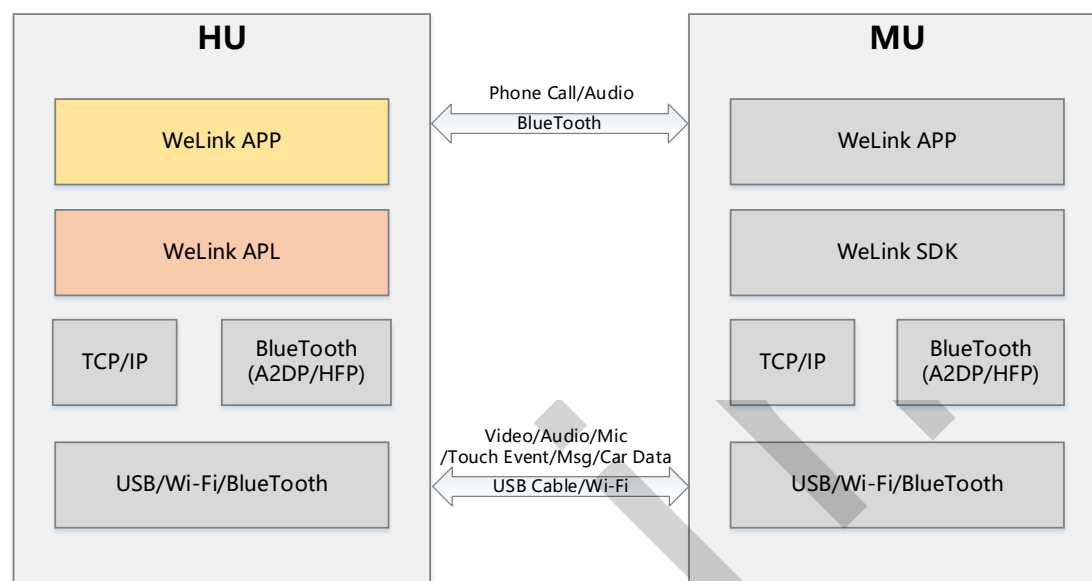


图 2-1 WeLink 方案整体框架

WeLink 方案的主要功能：是通过 USB/Wi-Fi/Bluetooth 等传输媒介（[2.1.1](#)），建立车载导航机与手机间的通信，通过数据流（[2.1.2](#)）的传递，实现二者间的智能连接。

2.1.1 传输媒介

WeLink 使用到的传输媒介包括：USB、Wi-Fi、Bluetooth 三种，对于单次手车互联来说，不需要同时使用这三种媒介，使用 USB + Bluetooth 或者 Wi-Fi + Bluetooth 即可。

- 1、USB：用作车机端与手机端通信的主要通道，支持的设备包括：Android(ADB/AOA)，iPhone(EAP)；
- 2、Wi-Fi：用作车机端与手机端通信的主要通道，支持的设备包括 Android，iPhone；
- 3、Bluetooth：主要用于传输手机端音乐数据以及电话语音（A2DP/HFP）。

2.1.2 数据流

车机与手机的通信数据主要分为：Video/Audio/Mic /Touch Event/Msg/Car Data/Phone Call 七种。

- 1、Video：是指手机端图像数据（手机端→车机端），传递方式：USB/Wi-Fi；

- 2、**Audio:** 是指手机端音频数据（手机端→车机端），包括微信、导航音、音乐、语音播报等，传递方式：USB/Wi-Fi/BlueTooth，这里需要注意：如果通过 USB/Wi-Fi 传递，需要车机端 WeLink APP 对收到的 Audio 数据进行解析、播放等处理；
- 3、**MIC:** 包括微信回复、以及语音识别等数据，MIC 录音有两种方式：车机端录音/手机端录音，如果选择手机端录音，则车机端不需要任何处理；如果选择车机端录音，则需要车机端根据具体的消息，进行录音操作，并将录音数据传递给手机端，传递方式：USB/Wi-Fi；
- 4、**Touch Event:** 是指触控数据（车机端→手机端），WeLink 连接成功后，用户可以通过点击车机端屏幕来操作手机，传递方式：USB/Wi-Fi；
- 5、**Msg:** 是指车机与手机端通信的消息数据（车机端↔手机端），用作参数传递，动作触发等，传递方式：USB/Wi-Fi；
- 6、**Car data:**是指车身自定义数据，是预留给车机端 APP 与手机端的通信数据通道，传递方式：USB/Wi-Fi；
- 7、**Phone Call:** 是指电话音频数据（手机端↔车机端），传递方式：BlueTooth。

2.2 WeLink 方案车机端架构

WeLink 车机端架构如图 2-2 所示。

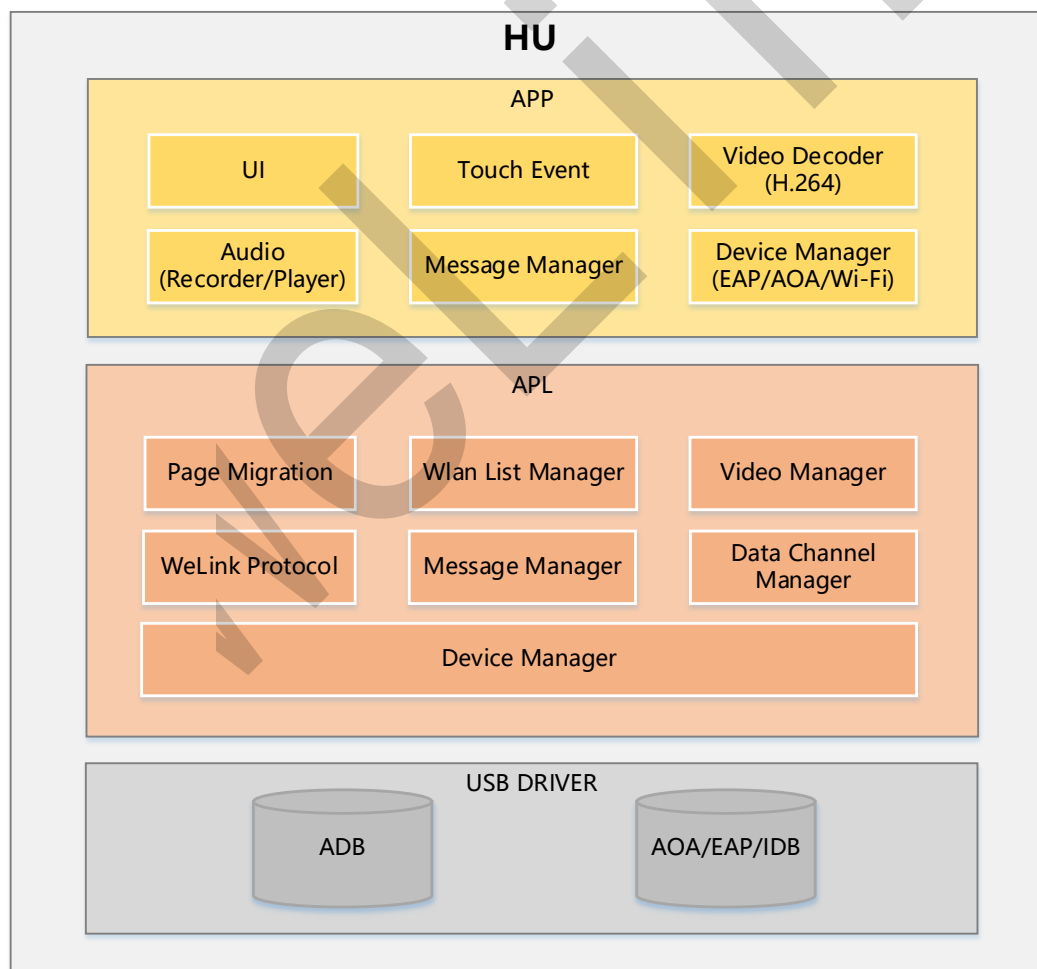


图 2-2 车机端架构

对于采用 WeLink APL 方案接入的车机厂商或供应商来说，需要完成上图中 APP 层的开发工作，而下面的 APL 和 USB Driver 则属于四维开发内容。其中，USB Driver 主要用于 USB 的数据通道控制，在此不做介绍，下面我们对 APL 和 APP 各自的功能职责进行详细描述。

2.2.1 WeLink APL 功能

WeLink APL 主要功能包括：

- 1、Page Migration: 页面迁移管理，APL 会根据手机端的状态，以及 APP 的消息来控制画面更新，APP 只需要根据 APL 的页面更新消息来描画即可；
- 2、Wlan List Manager: 无线列表管理，主要用于无线连接，判断是否有支持 WeLink 的手机连接到车机，如果有新的设备连接，会通知 APP；
- 3、Video Manager: 手机端图像数据管理，手机端传递到车机端的图像数据分为两种：JPG、H.264，APL 会对 JPG 数据进行解码，转换为 RGB 数据再传递到 APP，对 H.264 数据做简单的解析拼包处理，再传递到 APP；
- 4、WeLink Protocol: 主要负责与手机端通信协议的管理，包括协议解析和协议封装，APP 层不需关注；
- 5、Message Manager: 消息管理，主要负责对 APP 层的消息，以及手机端的消息进行管理，与 APP 交互的消息；
- 6、Data Channel Manager: 主要负责与手机通信的数据通道管理，主要包括：
 - 【图像数据通道】：用于传输手机端图像数据，以及车机端触控坐标数据，必需通道；
 - 【消息数据通道】：用于传输车机端与手机端的通信消息，必需通道；
 - 【音频数据通道】：用于传输手机端音频数据（音乐数据、语音播报数据），只有音频数据不走蓝牙，而是走 USB/Wi-Fi 时才需要建立此通道；
 - 【MIC 数据通道】：用来传递车机端 Mic 录音数据到手机端，只有在语音识别走车机端时才需要建立此通道；
 - 【车身数据通道】：此通道是车机 APP 与手机自定义的数据通道，APL 不做处理，只做转发，如果车机 APP 没有与手机自定义通信协议则不需要建立此通道；
- 7、Device Manager: 手机设备状态管理，其中 EAP/AOA/Wi-Fi 设备需要 APP 发送插入拔出通知，ADB 设备则由 APL 自己检测。

2.2.2 车机端 APP 需要对应的功能

- 1、UI: 界面显示功能，主要负责根据 APL 提供的页面状态（Page）来进行描画操作，具体页面参照[第 9 章](#)；
- 2、Touch Event: 触控管理，主要负责获取屏幕触控信息，并通知 APL，参照[第 7 章](#)；
- 3、Video Decoder: 视频解码，主要负责将手机端传过来的 H.264 图像数据解码上屏，参照[第 5 章](#)；
- 4、Audio: 音频管理，包括录音、音频播放（USB/Wi-Fi/BT），此模块需要根据具体项目定制，参照[第 6 章](#)；
- 5、Message Manager: 消息管理，主要负责与 APL 的消息通信，以及与车机系统的消息通信，参照[第 8 章](#)；
- 6、Device Manager: 设备管理，AOA/EAP/Wi-Fi 设备的插入和拔出需要在 APP 层管理，具体参照[第 4 章](#)；

车机端 APP 完成上面几个模块功能后，WeLink 程序即可正常运行。

至于上述之外的其他功能：下文（第 9 章之后）会有独立章节进行详细描述。

WeLink

3. 车机平台软硬件需求

分类		详细说明		
硬件	CPU	主频 800MHz 以上		
	图形	支持 OpenGL ES 或 DirectDraw 硬件加速		
	USB	1. 支持 USB 2.0 及以上 HOST 模式,并支持车机 USB 模式在 HOST 与 DEVICE 之间的动态切换。 2. Android 手机：工作在 Host 模式。 3. IOS 手机：EAP 方案，工作在 Device 模式。 4. USB 数据传输速率：最低 16,000kbit/s，理想传输速率 24,000kbit/s 及以上。 5. USB 稳定持续供电 5V 1.5A，数据传输 5V 500mA。		
	WIFI	HU 带有 WIFI 模块。 WIFI 在 2.4GHz 模式工作时传输会受到 BT 干扰，最恶劣情况个别手机开启 BT 会造成 WIFI 连接中断，故 HU 需解决 WIFI 与 BT 的冲突问题。 数据传输速率：最低 16,000kbit/s，理想传输速率 24,000kbit/s 及以上。		
	蓝牙	支持 A2DP 和 HFP		
	视频输出	支持 H264 视频解码和显示。 支持 H.264/AVC 的硬件解码，解码能力 high level 4 以上，分辨率为 1280*720/30 帧及 1920*1080/30 帧。 硬解码一帧数据耗费时间 <10ms。 支持硬件显示 YUV 数据，显示一帧 YUV 数据 <10ms。		
	内存需求	30M		
	ROM 需求	程序包放置空间	ADB 60M	AOA 6M
软件	OS	Linux : 2.6 或以上		
	驱动	USB HOST 驱动		
	H264	H264 硬解码 API		
	协议栈	蓝牙协议栈：HFP、A2DP TCP/IP 协议栈		

4. 连接方式

目前 APL 支持的连接方式包括：ADB、AOA、EAP、以及无线。下面分别介绍。

4.1 AOA 连接

4.1.1 AOA 简介

AOA（Android Open Accessory）是 Google 提供的 Android 设备与 Android 配件之间进行 USB 通信需遵循的连接协议，该协议对 Android 配件如何检测 Android 设备并如何与其建立通信作出了详细规定。

AOA 协议规定配件需要有以下功能：

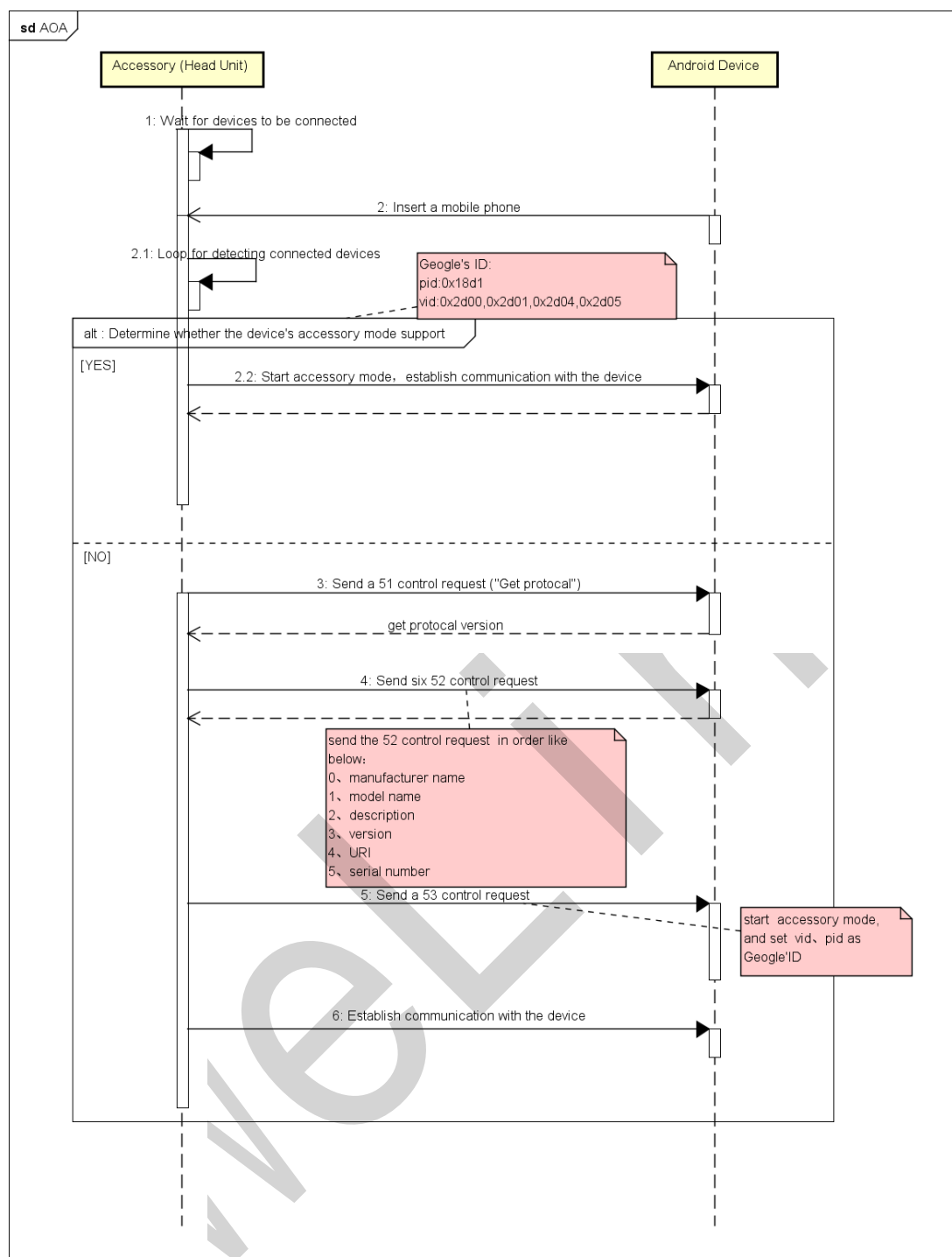
- （1）侦听自身的 USB 接口，等待与其连接的 Android 设备。
- （2）检测 Android 设备是否支持 AOA 协议。
- （3）如果需要，启动 Android 设备的 USB 配件模式。
- （4）与 Android 设备建立 USB 通信。

Android USB 通信模式，分为主机模式（Host Mode）和配件模式（Accessory Mode）两种模式。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；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。

对于 WeLink 来说，手机设备使用的是配件模式，车机端需要有操作 USB 端口的权限。

4.1.2 AOA 配件切换说明

AOA 配件切换交互流程图参照图 4-1。



powered by Astah

图 4-1 AOA 配件切换时序图

说明:

1. 配件端等待手机连接。
2. 手机接入配件。配件枚举连接过来的手机。通过枚举获取的 VID 是否为 0x18D1, 以及 PID 是否为 0x2D00、0x2D01、0x2D04 或 0x2D05, 判断手机是否处于配件模式, 如果是可跳到步骤 7, 否则尝试进入配件模式;

0x2D00 – accessory

0x2D01 – accessory + adb
0x2D02 – audio
0x2D03 – audio + adb
0x2D04 – accessory + audio
0x2D05 – accessory + audio + adb

3. 确定设备是否支持配件模式，发送 51 控制请求（“获取协议”）以确定设备是否支持 Android 配件协议。如果设备支持协议，则返回一个非零数字，代表所支持的协议版本；

requestType: USB_DIR_IN | USB_TYPE_VENDOR
request: 51
value: 0
index: 0
data: 协议版本号（16 位小端字节从设备到配件）

4. 如果设备返回所支持的协议版本，则向设备发送含标识字符串信息的控制请求 52。该信息让设备可以确定适合配件的应用（如果没有适合配件的应用，则向用户呈现一个网址）；

requestType: USB_DIR_OUT | USB_TYPE_VENDOR
request : 52
value : 0
index: string ID
data: 零终止的 UTF8 字符串从配件发送到设备
支持以下 string ID，每个字符串的最大大小为 256 字节（必须以\0 结尾）。
manufacturer name: 0 //制造商名称
model name: 1 //型号名称
description: 2 //描述
version: 3 //版本
URI: 4 //URL
serial number: 5 //序列号

5. 发送控制请求 53，要求设备以配件模式启动；

requestType: USB_DIR_OUT | USB_TYPE_VENDOR
request: 53
value : 0
index : 0
data: none

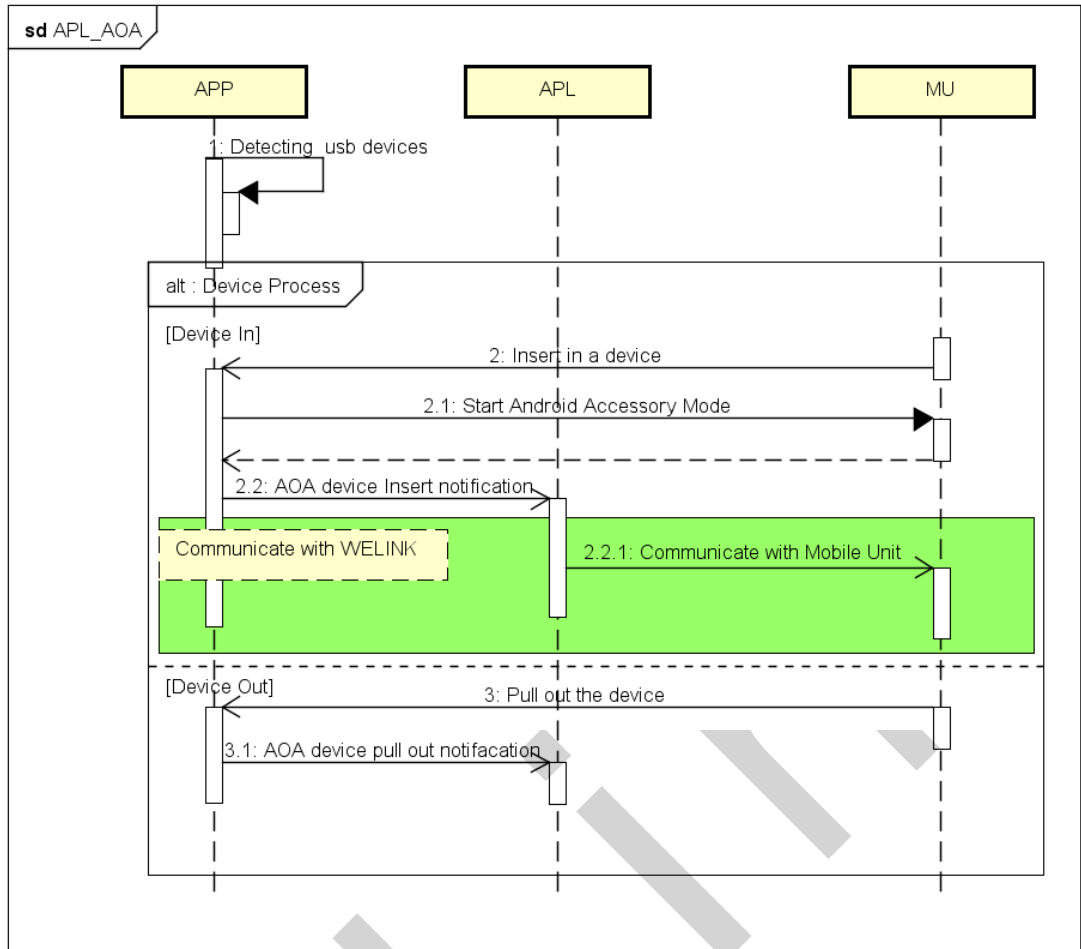
6. 完成这些步骤后，配件应等待所连接的 USB 设备在配件模式下将其自身重新接入总线，然后重新枚举所连接的设备；

7. 确定已经进入配件模式，重新枚举设备，配置 USB 端点和接口，与设备建立通信。

注：以上控制请求都是端点 0 上的请求。

4.1.3 AOA 通道的建立

通过 USB 方式 Android 手机与车机建立 AOA 通道的流程图如下所示：



powered by Astah

图 4-2 AOA 通道建立时序图

AOA 通道建立的整体流程

1. usb 设备检测；
2. AOA 配件切换；
3. AOA 设备插入，通过 APL 接口通知 AOA 设备插入；
4. 车机端 WeLink 与手机端进行互联通信；
5. 拔出手机，调用 APL 接口通知 AOA 设备拔出。

4.1.4 车机端的开发工作

由前文时序图可以看出，对于 AOA 互联方案，需要车机端与 WeLink APL 以及 WeLink 手机端共同配合，完成手机配件模式的切换以及通道的建立，下面重点强调一下车机端 APP 层开发人员需要对应的开发内容：

1. AOA 设备的识别；
2. 对手机设备进行 AOA 配件模式的切换，AOA 配件字符串信息请向 WeLink 团队索取。
3. 完成配件切换后，调用 APL 接口函数，通知 WeLink“设备接入”。
4. 当车机端识别到 AOA 设备断开 USB 连接，调用 APL 接口函数。通知 WeLink“设备拔出”。这里提到的 API 函数是 `wl_apl_device_chg()`，后文会有详细描述。

4.2 EAP 连接

4.2.1 EAP 简介

EAP，全拼是 External Accessory Protocol，外部设备协议，是苹果官方提供的外设和 iPhone 手机通信协议，这个是苹果推荐使用的外设连接方式。需要外设集成 MFI 芯片进行 MFI 认证。它有两种模式，一种是叫 EASession 的模式，它带宽相对较低，但是允许同时通过多个协议字符串创建多个会话，也就是说直接支持多个通道；另外一种 Native Transport 的模式，这种模式的优点是带宽足够大，但是不支持多通道，并且 Native Transport 需要 iPhone 工作在 USB host 模式，硬件需要支持 USB 模式切换。

4.2.2 车机端的开发工作

车机系统开发者需要对应的工作主要包括：MFI 认证、EAP 驱动并提供 EAP 数据读写方法。

而车机端 APP 层开发者需要注意 EAP 连接时 APP 层需要完成的任务：

1. iPhone 手机插入拔出的识别；
 2. role switch 以及 IAP2 认证；
 3. 完成 2 中事宜之后，调用 APL 接口函数，通知 WeLink“设备接入”。
 4. 当车机端识别到 AOA 设备断开 USB 连接，调用 APL 接口函数。通知 WeLink“设备拔出”。
- 关于 BundleID 以及 ExternalAccessoryName，请参照《WeLink 车机端 EAP 说明文档（车机端_Linux）》文中描述。
- 这里提到的 API 函数是 `wl_apl_device_chg()`，后文会有详细描述。

4.2.3 EAP 通道的建立

通过 USB 方式 iPhone 与车机 EAP 通道建立的流程图如下所示：

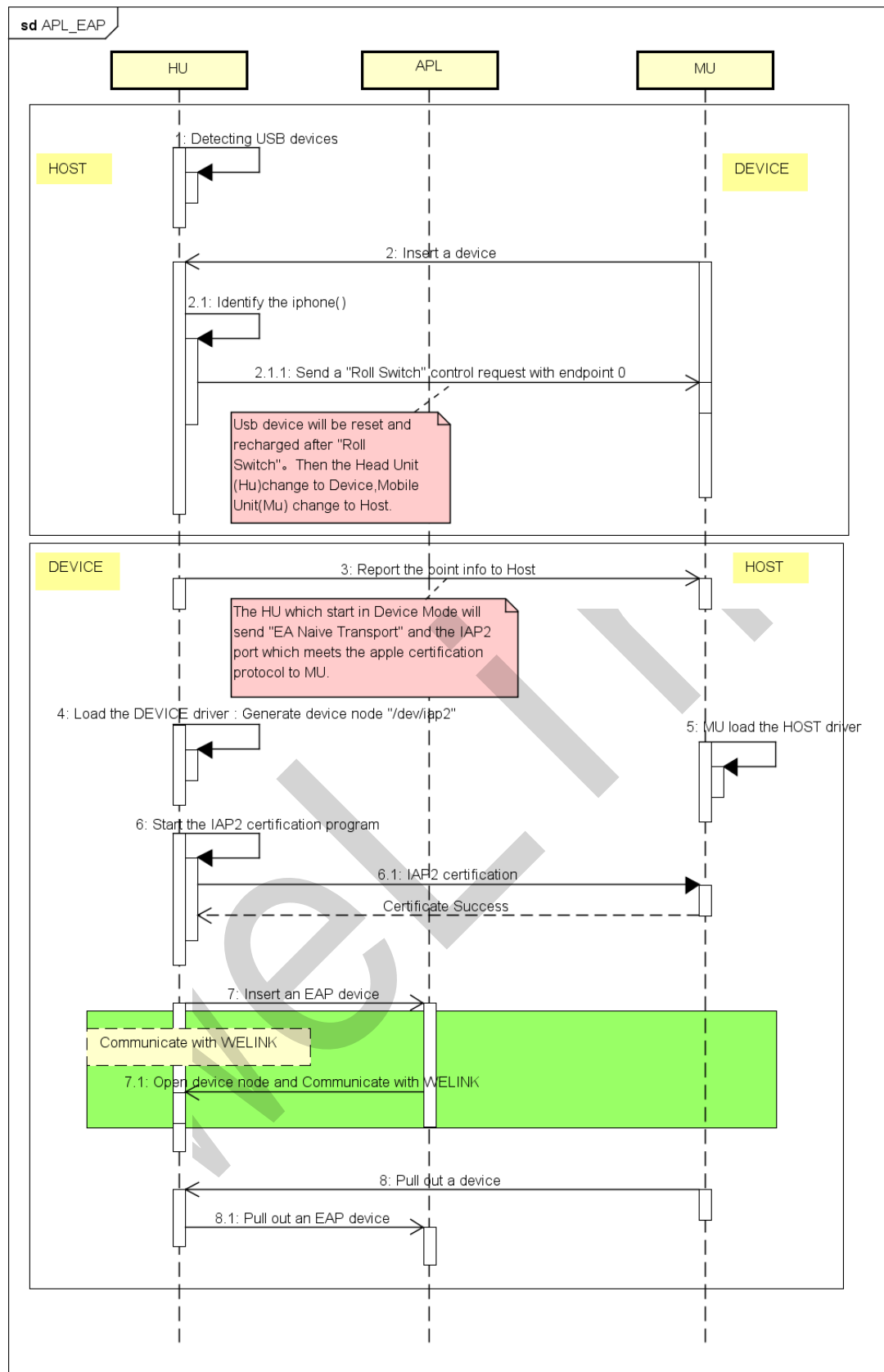


图 4-3 EAP 通道建立时序图

EAP 通道建立的整体流程：

1. 车机处于 HOST 模式，枚举手机
iPhone 手机（通过 USB 线）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 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/eaNative）、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/iap2）。
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 驱动映射的设备文件）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. 调用 APL 接口通知 EAP 设备插入。
10. 打开设备文件节点，与手机端进行数据交互完成 WeLink 连接。
11. 拔出 ios 手机，调用 APL 接口通知 EAP 设备拔出。

4.3 WIFI 连接

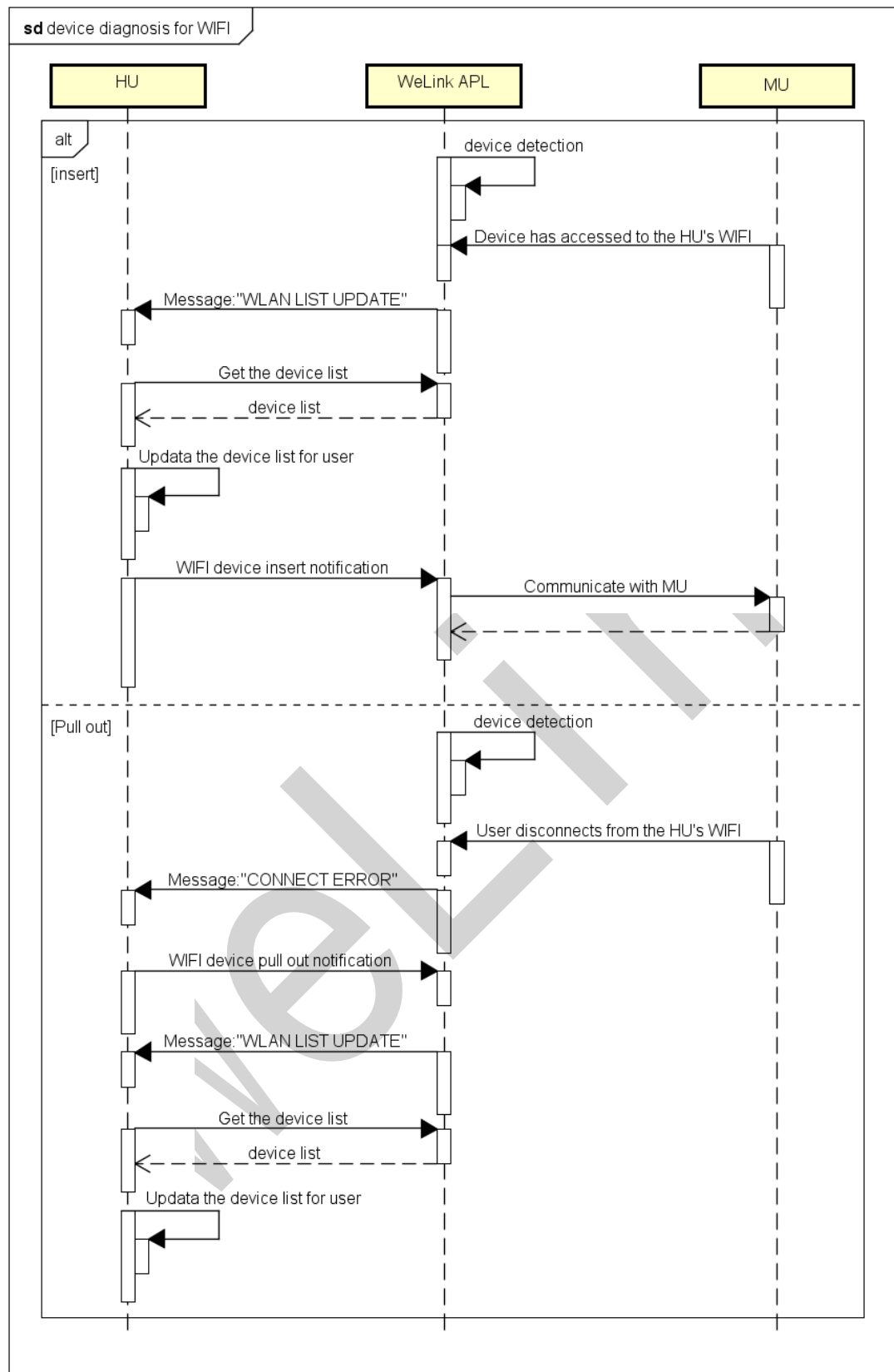
4.3.1 WIFI 机能简介

WeLink 除了支持 USB 互联，还支持 WIFI 无线互联。有车机做热点和手机做热点两种方式，具体使用哪种方式，需要车机厂商或供应商与 WeLink 团队沟通确认。

与 USB 互联不同的是，使用 WIFI 互联的前提是：手机与车机需要进行 WIFI 连接，并且手机上的 WeLink 应用需要打开（可以是多个手机）。此时，WeLink APL 会收集所有已经与车机 WIFI 连接并且打开 WeLink 应用的手机信息（IP、MAC 地址、名称），组成列表，向车机端 APP 发送列表更新的消息。车机端在收到此消息后，取得该列表，展现给用户。如果用户选择点击其中一台，则意味着该手机开始与车机进行 WeLink 互联。

4.3.2 WIFI 设备检测

WeLink WIFI 互联方案中，无线设备的检测时序图如下：



powered by Astah

图 4-4 WIFI 设备检测时序图

说明：

1. 设备列表更新机制

- APL 检测手机发过来的 UDP 广播消息；
- 解析UDP消息，如果是新设备，则查找该设备Mac地址；
- 找到Mac地址后，将该设备添加到设备列表中，并通知车机端APP层更新设备列表；
- 对设备列表中的设备进行断开连接监测：
 - A) 对于已经和车机WeLink互联的设备，只有在设备和车机WeLink断开连接的时候，从设备列表中清除，并通知APP更新设备列表。
 - B) 对于没有和车机 WeLink 互联的设备，10 秒之内收不到设备的 UDP 消息，认为该设备已经断开 wifi 连接，从设备列表中清除，并通知 APP 更新设备列表。

2. 选择手机设备进行互联

车机端 APP 判断如果车机和某一手机设备已经处于连接中的状态，那么在连接新设备之前，必须先（调用 APL 接口）通知 WeLink APL“设备拔出”，然后再（调用 APL 接口）通知 APL 新“设备接入”。

3. 异常处理

车机和手机设备连接的过程中，出现的异常断开（WIFI 断开、手机 APP 退出、手机关机等）的场合，APL 会向车机端 APP 反馈消息“连接失败”，APP 接收到该消息后通知 APL“设备拔出”。

4.3.3 车机端的开发工作

车机系统开发者需要实现车机的 WIFI 连接功能。

而车机端 APP 开发者需要根据上面时序描述的内容，完成设备接入的一系列动作，需要注意的是：

1. 这里提到的“设备接入/拔出”的 API 函数仍然是 `wl_apl_device_chg()`；
2. 取得无线设备列表的 API 函数是 `wl_apl_get_wlan_list()`；
3. 车机端在得到“无线设备列表更新”的消息后，需要将列表内容展示给用户，以供用户选择将要使用 WeLink 的设备。展现的方式是车机端 HMI 需要考虑的。

4.4 ADB 连接

4.4.1 ADB 简介

ADB（Android Debug Bridge）是 Google 提供的与 Android 设备进行调试的通信协议，与其他连接方案不同的是，ADB 设备的识别，由 APL 实现，对 APP 来说，不需要关注具体的协议内容。

4.4.2 车机端的开发工作

针对 ADB 方案，车机端 APP 不需要关注设备的插拔识别，APL 识别到 ADB 设备后，会自动进行连接操作，所以 APP 只需要处理连接之后的动作，与其他连接方案都是共通的。

4.5 WeLink 连接时序

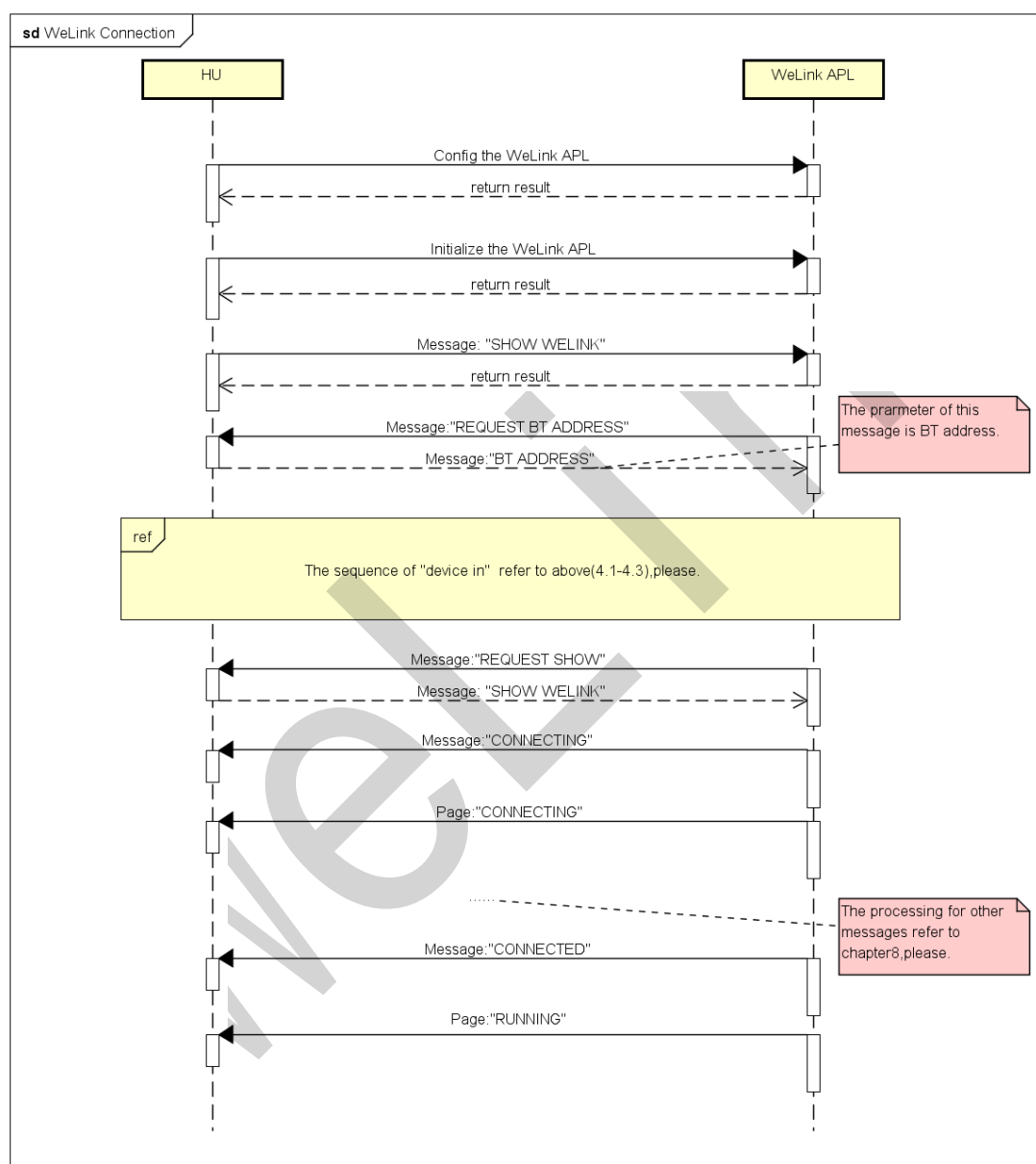


图 4-5 WeLink 连接时序图

- 1、配置并初始化 APL，参照 [13.2.1](#)；
- 2、切换 WeLink 到前台；
- 3、APL 获取车机端蓝牙地址，用于做蓝牙自动连接，不需要自动连接蓝牙的项目可以忽略；
- 4、设备识别，参照 [第四章](#)；
- 5、APL 开始进行连接，连接过程中会与上层 APP 有消息与页面交互；

6、连接成功后，通过页面回调函数通知上层进行页面描画；

如上图可见，WeLink 互联的整个过程，除了本章前几小节介绍的设备识别和数据通道建立之外，基本就是消息（Message）和页面（Page）的处理，这两部分涉及到 WeLink 业务逻辑，需要车机端开发者重点关注，后文会有独立章节详细介绍。



5. 视频引擎

5.1 视频流

关于视频流数据，作用是将手机端图像数据投影到车机端，APP 需要根据 APL 的页面状态，以及 APL 的图像数据更新，来控制图像数据的解码、以及描画显示。

■相关接口函数：

`video_data()`：传递手机端图像数据，APP 收到数据后需要进行解码操作；

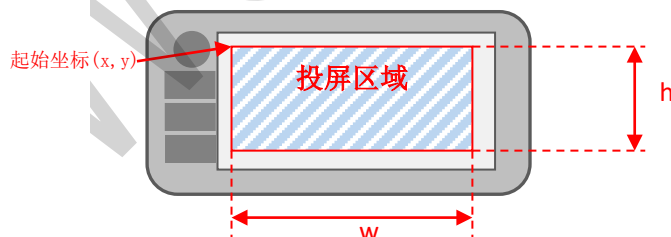
`update_page()`：APL 当前页面状态通知，只有在 `WL_PAGE_TYPE_RUNNING` 状态下，APP 才需要显示解码后的图像；

■APP 解码、描画显示的控制：

- (1) 初始化解码器：初次连接成功、或者连接成功后车机端 WeLink 由后台切换到前台时，APP 会先收到 `WL_PAGE_TYPE_RUNNING` 消息，之后才会收到 `video_data()` 回调的图像数据，APP 可以在收到数据时初始化解码器（因为图像数据中有宽、高信息，初始化解码器的时候可能会用到）；
- (2) 解码过程中：连接成功后，APP 需要将 `video_data()` 回调的数据正常向解码器中放，但是，只有在页面状态是 `WL_PAGE_TYPE_RUNNING` 时，APP 才可以显示解码后的图像；
- (3) 释放解码器：APP 在收到 `WL_APL_MSG_TYPE_APL_CONNECT_ERR` 或者 `WL_APL_MSG_TYPE_APL_DISCONNECT` 消息时，需要释放解码器；另外，车机端 WeLink 由前台切换到后台之后，APP 会收到 `WL_PAGE_SOURCE_OFF` 页面更新消息，此时，APP 也需要释放解码器；

5.2 屏幕显示区域的设定

对 WeLink APL 初始化之前，需要配置 APL 参数 `wl_apl_screen_position_t`（配置的方法示例后文有详述和举例），其中就有车机端屏幕投屏显示区域的设定，APL 会根据此区域来转换车机端 APP 发送的触控坐标，让用户在车机端的触控可以正确的操作到手机。



6. 音频引擎

6.1 音频播放

手机端的音频播放可以根据项目定制，选择走 Bluetooth 通道，或者 USB/ Wi-Fi 通道。其中，Bluetooth 通道不做详细介绍，下面着重介绍一下 USB/Wi-Fi 通道传输音频的方案。

首先，车机厂商或者供应商向 WeLink 团队索取《WeLink 音频数据通信协议》文档，理解协议文档后，与 WeLink 手机端围绕音频管理进行讨论沟通、确认需求。

对于 USB 音频这个机能，WeLink APL 所起的角色是“管道”，WeLink 手机端应用按照既定协议，将 pcm 数据发送到 APL，APL 不做任何处理（也不做缓存），直接将 pcm 数据包通过回调（callback）方式反馈给车机端 APP 层。具体协议参照《WeLink 音频数据通信协议》文档。

■ 相关函数接口：

audio_data(); 回调函数，APL 通过此函数传递 PCM 数据给 APP；

wl_apl_send_audio(); 通信接口，APP 通过此接口回复 PCM 相关协议数据。

车机端在收到回调的 pcm 数据后，需要按照《WeLink 音频数据通信协议》进行数据解析，并及时取出 pcm 数据包，完成播放，避免阻塞音频数据通道，并尽量减少延迟。关于各音源的管理、缓存、混音以及音源之间的打断，还有车机系统音频设备的资源管理都需要车机厂商或者供应商（车机端 APP 层）来统一处理。

6.2 MIC 录音

WeLink 互联成功后，可以在【设置】界面中选择【手机录音】或是【车机录音】。

- 手机录音：使用手机自身的 Mic，车机端无需打开车机 Mic。
- 车机录音：使用车机 Mic 进行录音，互联成功后 APL 会向车机端 APP 层发送“开始录音”消息 WL_APL_MSG_TYPE_APL_START_RECORD，车机端 APP 在接收到此消息时，打开车机 Mic 并实时采集 Mic 数据（pcm），通过 wl_apl_send_record_data()接口函数将 pcm 数据持续发送给 WeLink。
同理，当车机端 APP 收到“结束录音”消息 WL_APL_MSG_TYPE_APL_STOP_RECORD 时，需要关闭车机 Mic。

Note: 由于 WeLink 支持实时唤醒，所以如果选择车机录音（互联成功后），会立即收到开始录音的消息。当然，对于某些项目，如果采用硬按键方式来触发“开始录音”也是可以的，具体需要与 WeLink 团队沟通讨论。

7. 触控（Touch）

7.1 车机端的开发工作

如果车机屏幕是触摸屏，那么车机端只需要将车机系统的触控事件（Down、Move、Up），触控坐标（x, y），通过接口函数发送给 WeLink 即可实现回控。

WeLink APL 触控接口分为两种：

1、单点触控：

对应的接口函数：wl_apl_point_down()、wl_apl_move()、wl_apl_point_up()。

2、多点触控：

对应的接口函数：wl_apl_send_point()、wl_apl_send_point_sync()。

需要注意：

- (1) 两种接口不能同时使用，同一个项目只能选择支持单点触控或者多点触控；
- (2) 调用触控接口，需要在车机端正式投影手机端图像之后；

7.2 触控坐标

为了保证连接成功后，在车机端的触控操作可以正确的控制手机，APP 需要完成：

- 1、对 APL 初始化时正确设置车机投影区域；
- 2、按照车机端的坐标系来传递坐标；

关于车机端到手机端的坐标换算，已经封装到 APL 中，车机端 APP 无需关注。

8. 消息通信（Message）

WeLink 的状态与控制，是通过消息（Message）交互（控制流）的方式来实现的。

WeLink 向车机端发送的消息，通过回调函数 recv_msg()反馈到车机端 APP 层；

而车机端向 WeLink 发送的消息，通过接口函数 wl_apl_send_msg()发送给 WeLink APL。

8.1 APL 反馈给车机端 APP 的消息

- WL_APL_MSG_TYPE_APL_REQ_SHOW
请求显示 WeLink 消息，通知车机端 APP 此时需要将车机 WeLink 置于前台显示。
- WL_APL_MSG_TYPE_APL_REQ_HIDE
请求隐藏 WeLink 消息，通知车机端 APP 此时需要将车机 WeLink 置于后台显示。
- WL_APL_MSG_TYPE_APL_OPEN_BT
请求打开蓝牙，APP 收到该消息，需要打开蓝牙处理，如果已经打开，不需操作。
暂时不需关注。
- WL_APL_MSG_TYPE_APL_BT_STATUS
通知车机端 APP 此时手机端的蓝牙状态，参数为蓝牙状态：1：已连接，0：未连接。

暂时不需关注。

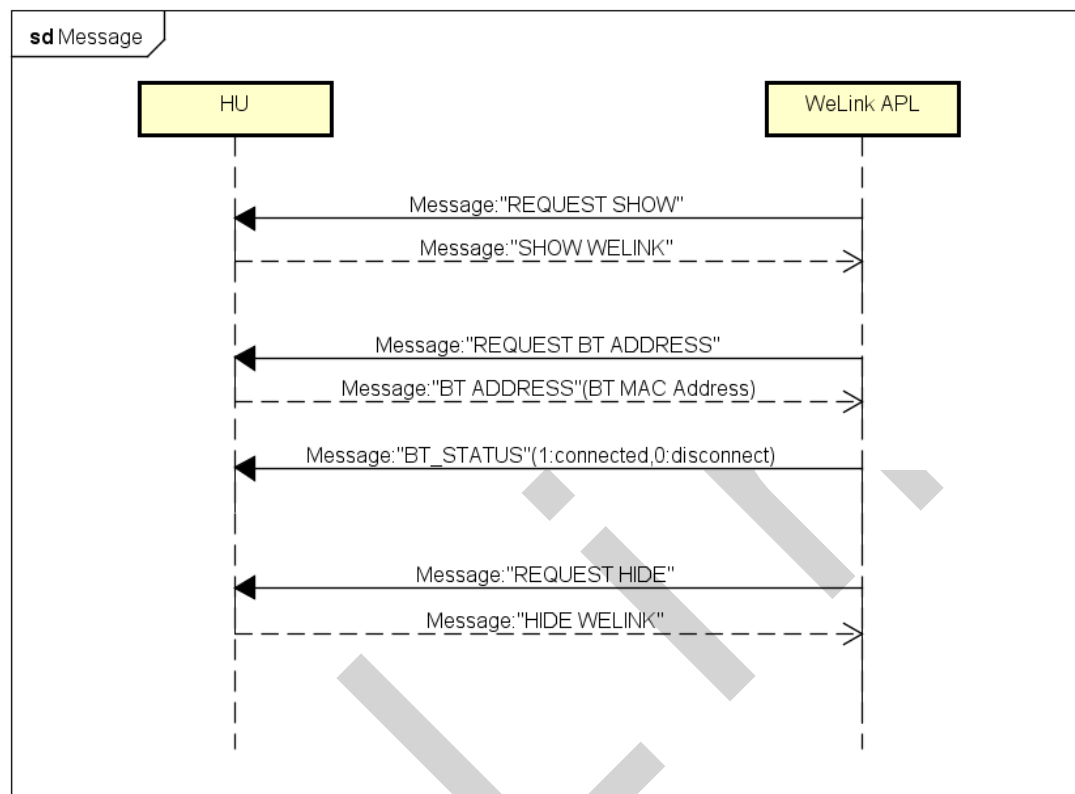
- **WL_APL_MSG_TYPE_APL_REQ_BT_ADDR**
请求发送蓝牙地址，车机端 APP 收到该消息，向 APL 发送车机蓝牙 MAC 地址。
- **WL_APL_MSG_TYPE_APL_RECV_MSG**
车机端 APP 收到该消息，需要根据消息的参数，闪烁消息图标。
参数说明：1:微信，3:其他
- **WL_APL_MSG_TYPE_APL_NAVI_GUIDE_START**
开始显示导航条信息（仅供有导航条功能的项目使用）。
- **WL_APL_MSG_TYPE_APL_NAVI_GUIDE_UPDATE**
更新导航条信息（仅供有导航条功能的项目使用）。
- **WL_APL_MSG_TYPE_APL_NAVI_GUIDE_STOP**
开始显示导航条信息（仅供有导航条功能的项目使用）。
- **WL_APL_MSG_TYPE_APL_UPDATE_KEY**
车机端 APP 收到该消息，需要根据参数，将当前 Key（特指常驻条上的 Button，取值从 WL_PAGE_BTN_LAUNCHER 到 WL_PAGE_BTN_MIC）设置成高亮显示。
参数的值就是当前需要高亮显示的常住条 Button 的 Key 值。
- **WL_APL_MSG_TYPE_APL_CONNECTING**
通知车机端 APP，此时车机和手机设备处于 WeLink 互联中状态。
- **WL_APL_MSG_TYPE_APL_CONNECTED**
通知车机端 APP 此时车机与手机设备 WeLink 互联成功。
- **WL_APL_MSG_TYPE_APL_CONNECT_ERR**
连接失败消息，用于通知车机端 APP，此时手机与车机连接发生了异常断开，需要调用 APL 接口函数通知 APL“设备断开”。
- **WL_APL_MSG_TYPE_APL_DISCONNECT**
通知车机端 APP 此时车机与手机设备已经断开 WeLink 互联。
- **WL_APL_MSG_TYPE_APL_START_RECORD**
开始录音消息，车机端 APP 收到该消息，需要打开车机 Mic 开始录音。
- **WL_APL_MSG_TYPE_APL_STOP_RECORD**
结束录音消息，车机端 APP 收到该消息，需要关闭车机 Mic 结束录音。
- **WL_APL_MSG_TYPE_APL_WLAN_LIST_UPDATE**
无线设备列表更新消息，用于通知车机端 APP 无线设备列表发生了更新，车机端收到该消息后，需要从 APL 获取无线设备列表，用户可以选择其中某个设备进行互联。

8.2 APP 通知 WeLink APL 的消息

- **WL_APL_MSG_TYPE_APP_SHOW_WELINK**
向 APL 发送该请求，通知 WeLink 置于前台显示。
- **WL_APL_MSG_TYPE_APP_HIDE_WELINK**
向 APL 发送该请求，通知 WeLink 置于后台。
- **WL_APL_MSG_TYPE_APP_BT_ADDR**
向 APL 发送车机蓝牙 MAC 地址。蓝牙 MAC 地址作为该消息的参数。
- **WL_APL_MSG_TYPE_APP_SET_LANGUAGE**
向 APL 发送该消息，用于修改车机和手机设备当前显示语言。
暂时不需关注。

- WL_APL_MSG_TYPE_APP_SET_DRIVE_STAT
向 APL 发送该消息，用于通知车机和手机设备当前行驻车状态。
暂时不需关注。

8.3 消息交互时序图



powered by Astah

图 8-1 部分消息交互时序图

9. 页面简述（Page）

WeLink 使用过程中，除了展现手机投屏画面之外，在某些场合，还需要车机端 APP 描画相应的页面，来给用户展现其需要关注的提示信息。

这里有两部分内容，需要车机端 APP 开发人员关注：

1. 目前处于什么场景，需要描画什么页面？

这一点 APL 已经做了管理，将所有需要展现的页面定义为一组页面消息（注意这组消息是页面消息，文中记做 **Page**，不是前文所述的控制流消息 **Message**），通过回调的方式（回调函数 `update_page()`），将当前所需的页面描画处理通知给车机端 APP。车机端 APP 只需在收到描画消息之后，做相应的页面描画处理即可。

2. 页面上的 button

众所周知，某些页面是需要配有相应 button 的（例如退出当前页面），这里 WeLink 也都做了相应的规范和键值定义。当然，这些 button 的描画是需要车机端 APP 层完成，当用户点击了 button 后，车机端 APP 只要将点击的键值通过接口函数 `wl_apl_send_button()` 通知 APL 即可。

下面详细说明一下各个页面以及页面上的 button。

9.1 车机端需要描画的页面

本节先按照“页面消息（Page）”逐条介绍页面（宏定义参照参照 [12.4.5](#)），页面中用到的 button 在下一节统一介绍。

注意：这里的页面图片只是给出示例，具体页面风格、展示内容以及哪些页面可以合并和删减，请根据项目情况，与 WeLink 团队沟通讨论确定。

1. **WL_PAGE_SOURCE_OFF**

此时车机端 WeLink 程序隐藏（处于后台），车机端请显示车机系统自身画面。

2. **WL_INITIAL_PAGE**

未启动手机端 WeLink 或者未开始互联场景下，车机端可以显示该页面。车机端可以根据 HMI 设计判断是否可以删减。



3. WL_HELP_PAGE_ANDROID

Android 手机连接帮助界面，车机端可以显示该页面。车机端可以根据 HMI 设计判断是否可以删减。



4. WL_HELP_PAGE_IOS

苹果手机连接帮助界面，车机端可以显示该页面。车机端可以根据 HMI 设计判断是否可以删减。



5. WL_CONNECTING_PAGE_OFFICIAL_ACCOUNT

WeLink 公众号二维码画面。



6. WL_CONNECTING_PAGE

互联过程中的页面。



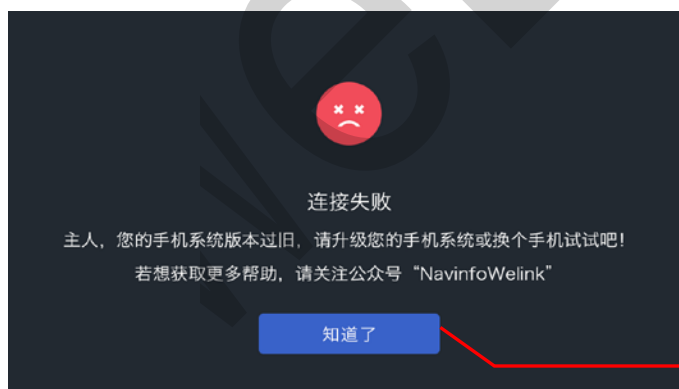
7. WL_CONNECTING_PAGE_AUTH

互联过程中权限验证的画面



8. WL_ERR_PAGE_SDK

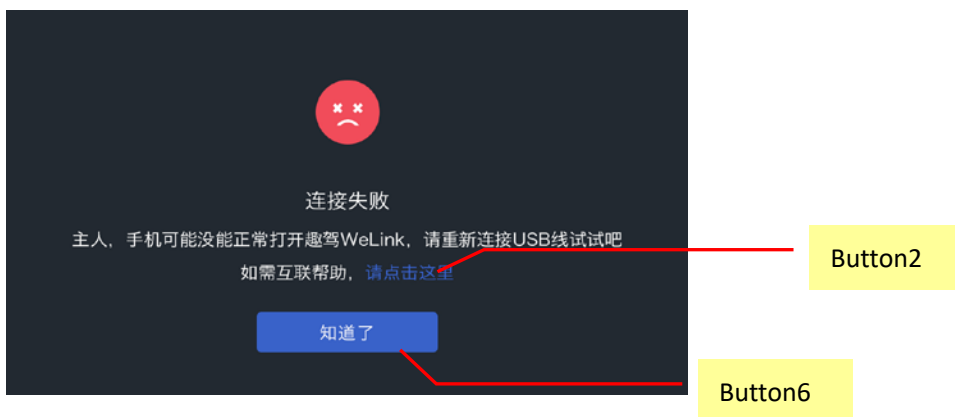
Android 手机系统版本过低导致 AOA 互联失败场景下，车机端可以显示该页面。车机端可以根据 HMI 设计将各种互联失败的页面进行合并管理。



Button6

9. WL_ERR_PAGE_CONNECT

AOA 互联失败场景下，车机端可以显示该页面。车机端可以根据 HMI 设计将各种互联失败的页面进行合并管理。



10. WL_ERR_PAGE_AUTH

互联过程中授权失败的显示页面。车机端可以根据 HMI 设计将各种互联失败的页面进行合并管理。



11. WL_ERR_PAPGE_AUTH_OVERDUE

互联过程中授权过期的显示页面。车机端可以根据 HMI 设计将各种互联失败的页面进行合并管理。



12. WL_ERR_PAGE_AUTH_NO_NETWORK

互联过程中离线授权失败的显示页面，车机端可以根据 HMI 设计将各种互联失败的页面进行合并管理。



13. WL_ERR_PAGE_HEART

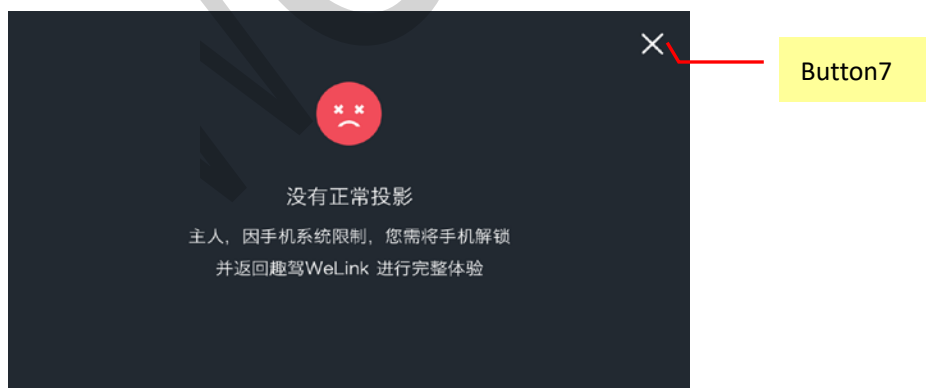
手机端 WeLink 应用被 kill 或者卸载，导致互联失败的场景，车机端可以显示该页面。车机端可以根据 HMI 设计将各种互联失败的页面进行合并管理。



14. WL_RUNNING_PAGE_LIMIT

WL_RUNNING_PAGE_SCREEN_OFF

互联后手动将 iPhone 手机端 WeLink 切换到后台及手机锁屏的场景，车机端可以显示该页面。



15. WL_RUNNING_PAGE_WECHAT_QRCODE

该消息表明用户在车机屏幕选择了微信功能，此时车机端 APP 需要描画一个二维码页面，来供用户手机扫描登录微信。这个二维码图片数据是通过回调函数的参数传递给车机端 APP 的。

车机端 APP 开发者需要注意：二维码旁边需要有一定的文言描述，来提醒用户扫码登录。



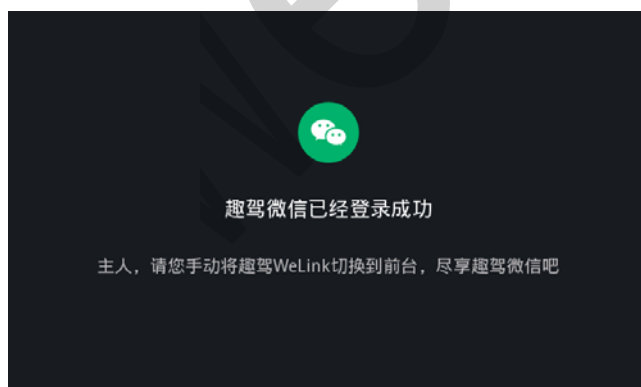
16. WL_RUNNING_PAGE_WECHAT_CONFIRM_LOGIN

用户微信扫描成功时，车机端可以显示该页面。



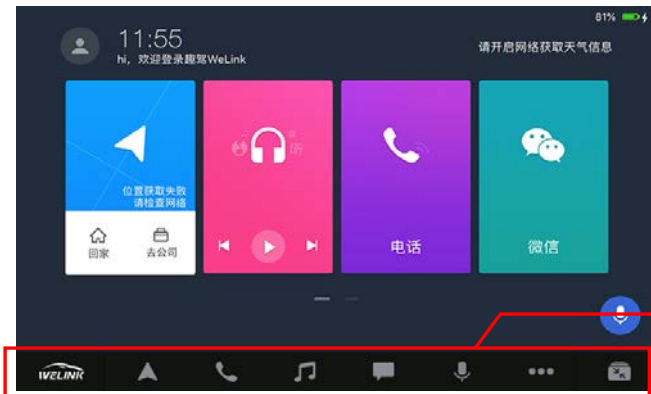
17. WL_RUNNING_PAGE_WECHAT_LOGIN_SUCCESS

微信登录成功后，车机端可以显示该页面。



18. WL_RUNNING_PAGE

这个页面消息比较特殊，当车机端 APP 回调函数中接收到该页面消息的时候，回调函数的参数数据代表的是手机的投屏数据（H264 或 JPEG）。这条消息的含义就是“请展现手机屏幕”。也就是说，在互联成功后（绝大多数时间里），只要需要展现手机投屏内容的时候，车机端 APP 都是在处理该条消息。具体内容前文第 5 章视频引擎已经详尽描述。另外，需要车机端 APP 开发者注意的是，车机屏幕下方的一排按钮，WeLink 团队习惯称之为“常驻条”。常驻条并非手机投影，而是需要车机端 APP 层描画的。具体需要哪些按钮，请每个项目根据自己的需求确定后，与 WeLink 团队沟通明确方案。



依次为 Button10 - 17

注:以下画面 AOA 方案暂时不需要关注

(1) WL_INITIAL_PAGE_NOT_AUTH

ADB 方案中，手机 USB 调试未开启提示画面

(2) WL_CONNECTING_PAGE_INSTALL

ADB 方案中，APP 安装中画面

(3) WL_ERR_PAGE_SERVICE

ADB 方案中，小服务启动失败画面

(4) WL_ERR_PAGE_CONNECT_ADB

ADB 方案中，初次连接失败画面

(5) WL_ERR_PAGE_INSTALL

ADB 方案中，安装 APP 失败提示画面

(6) WL_RUNNING_PAGE_SAFE_DRIVING

走行规制画面


(7) WL_RUNNING_PAGE_WAIT_FOR_VIDEO

等待图像数据更新

9.2 页面上的 button

前面介绍各个页面的同时，将页面上所需描画的 button 进行了标记。APL 对这些 button 已经做了定义，车机端 APP 层需要做好相应的 button 描画和高亮显示管理，当在判断用户点击动作之后，调用 APL 接口函数 `wl_apl_send_button()` 将相应的键值发送给 APL 即可，接下来的“跳转”动作，WeLink 会自行完成（宏定义参照 [12.4.6](#)）。

Button	图标示例	APL 中定义的键值	WeLink 动作
1		WL_PAGE_BTN_NONE_EXIT	退出 WeLink 回到车机画面
2		WL_PAGE_BTN_CLICK_HERE	进入帮助画面
3		WL_PAGE_BTN_HELP_BACK	返回到前一画面
4		WL_PAGE_BTN_HELP_ANDROID	进入安卓手机帮助界面
5		WL_PAGE_BTN_HELP_IOS	进入苹果手机帮助界面
6		WL_PAGE_BTN_BTN_GOT_IT	退出 WeLink 回到车机画面
7		WL_PAGE_BTN_LIMIT_CLOSE	退出 WeLink 回到车机画面
8		WL_PAGE_BTN_WECHAT_QRCODE_CLOSE	发送消息给手机端，恢复投屏数据的发送
9		WL_PAGE_BTN_WECHAT_CANCEL_LOGIN	恢复投屏数据的发送
10		WL_PAGE_BTN_LAUNCHER	切换到 WeLink 的 Home 页面
11		WL_PAGE_BTN_NAVI	切换到导航
12		WL_PAGE_BTN_PHONE	切换到电话
13		WL_PAGE_BTN_MUSIC	切换到音乐

14		WL_PAGE_BTN_MIC	切换到录音功能
15		WL_PAGE_BTN_MSG	切换到消息（微信）
16		WL_PAGE_BTN_MORE	切换到 MOS 账号
17		WL_PAGE_BTN_HOME	回到车机系统画面
18		WL_PAGE_BTN_OFFICIAL_ACCOUNT_SKIP	跳过该画面，开始进行手车互联
19		WL_PAGE_BTN_ERR_CLOSE	退出 WeLink 回到车机画面

如果需要追加或者删减某些 button，可以与 WeLink 团队沟通讨论并制定方案。

10. 蓝牙

与 WeLink 有关的蓝牙功能主要是 1. 蓝牙电话，2. 车载蓝牙音箱播放 Media 数据。

这就要求用户在使用 WeLink 时，需要保证手机和车机蓝牙的正确连接。WeLink 蓝牙的连接方式分两种，通过配置文件 wl_info.ini 中的 WL_AUTO_BT 选项控制，0 为手动连接，车机端 APP 不做任何处理，用户在使用 WeLink 时，自己手动将手机与车机的蓝牙进行配对连接；1 为自动连接，车机端 APP 需要在接收到 APL“请求蓝牙地址”消息后，将车机蓝牙地址（通过 *_APP_BT_ADDR 消息）发送给 WeLink，然后由 WeLink 手机端自行发起连接。

具体过程如下：

1. APL 初始化时，会发送 *_APL_REQ_BT_ADDR 消息请求蓝牙地址；
2. APP 收到 APL 发送的“请求蓝牙地址”消息后，通过 *_APP_BT_ADDR 消息将蓝牙地址发送给 APL，如果 APP 不发送，APL 默认蓝牙地址为“FF:FF:FF:FF:FF:FF”；
3. APL 保存车机蓝牙地址，在与手机端建立连接时，会将此蓝牙地址发送给手机端；
4. WeLink 手机端收到蓝牙地址会自动与之建立蓝牙连接。

11. 硬按键

关于硬按键功能，跟每个项目的具体需求关系紧密。因此需要车机厂商或供应商在明确具体需求之后，与 WeLink 团队详细沟通说明后，再来确定 APL 中需要提供哪些接口（API）和消息（Message）。

12. APL 接口说明

APL 的接口定义在 `wl_api.h` 中，其中有部分接口属于定制需求，需要根据具体项目需求来确定是否支持，如果项目不支持相关功能，则对应函数不可使用，具体参照表 12-1。

表 12-1 APL 接口函数

名称	功能	说明	备注
<code>wl_apl_set_config</code>	配置 APL 参数	共通	无
<code>wl_apl_init</code>	初始化 APL	共通	无
<code>wl_apl_deinit</code>	退出 APL，释放资源	共通	无
<code>wl_apl_point_down</code>	触控坐标压下（单点）	共通	无
<code>wl_apl_point_up</code>	触控坐标压下（抬起）	共通	无
<code>wl_apl_point_move</code>	触控坐标压下（移动）	共通	无
<code>wl_apl_send_button</code>	发送页面 button	共通	无
<code>wl_apl_send_msg</code>	给 APL 发送消息	共通	无
<code>wl_apl_get_version</code>	获取 APL 版本号	共通	无
<code>wl_apl_send_record_data</code>	发送 MIC 录音数据	项目定制	无
<code>wl_apl_send_audio_data</code>	用于 USB 传输音频时，车机与手机进行协议通信，具体协议参照《WeLink 音频数据通信协议》	项目定制	无
<code>wl_apl_send_point</code>	发送触控坐标（多点）	项目定制	无
<code>wl_apl_send_point_sync</code>	一次触控操作完成（多点）	项目定制	无
<code>wl_apl_send_hard_key</code>	发送硬件 KEY	项目定制	无
<code>wl_apl_device_chg</code>	通知 APL 设备插拔	项目定制	无
<code>wl_apl_get_wlan_list</code>	获取当前无线列表	项目定制	无
<code>wl_apl_send_hu_data</code>	发送车身 CAN data 数据	项目定制	无
<code>update_page</code>	页面更新回调函数	共通	无
<code>recv_msg</code>	APL 消息回调函数	共通	无
<code>audio_data</code>	USB 音频数据回调函数	项目定制	无
<code>hu_data</code>	车身数据通信回调函数	项目定制	无
<code>video_data</code>	手机端图像数据回调函数	共通	无

除了表 12-1 中列的函数外，APL 与 APP 的通信消息也有部分需要根据项目定制（参照 [12.4.4](#)），下面分类进行详细说明：

12.1 API

12.1.1 wl_apl_set_config

【原型】`int wl_apl_set_config (wl_apl_config_t *config)`

【功能】配置 APL 相关参数

【参数】

config: 设定 APL 配置信息，具体参照 [12.3.6](#)

【返回值】成功则返回 0，否则失败

【说明】同步函数，需要在使用 APL 其他函数之前调用此函数

【举例】

```
wl_apl_config_t config;  
//SSL 加密证书数据，暂时不需关注  
config.ssl_file.ca_file = NULL;  
//竖屏全屏显示，暂时不需关注  
config.hu_vertical_full = 0;  
//配置文件 wl_info.ini 文件路径，设置为 NULL，则默认获取当前路径  
config.config_file_path = NULL;  
//配置车机端版本号，APL 会发送给手机端  
config.huVersion = (char *)"V1.6.01709211";  
//授权码，需要根据具体项目定制  
config.serialNum = (char *)"FF:FF:FF:FF:FF";  
//设置车机 APP 支持的 RGB 描画格式  
config.fmt = WL_APL_IMAGE_FORMAT_RGBA;  
//设置屏幕位置（手机端图像投影区域）  
config.screen.x = 0;  
config.screen.y = 0;  
config.screen.w = 800;  
config.screen.h = 430;  
//设置消息回调函数  
config.cb_func.func_msg_notice = _p_recv_msg;  
//设置页面更新回调函数  
config.cb_func.func_update_page = _p_update_page;  
//设置 audio 数据回调函数，具体协议需要与手机端定义  
config.cb_func.func_audio_data = _p_audio_data;  
//车身数据回调函数，暂时不需关注  
config.cb_func.func_hu_data = _p_hu_data;  
//设置 APL config  
wl_apl_set_config(&config);
```


12.1.2 wl_apl_init

【原型】`int wl_apl_init(void)`

【功能】初始化 APL，启动 APL 工作主线程

【参数】无

【返回值】成功则返回 0，否则失败

【说明】同步函数

【举例】

```
//初始化 APL
wl_apl_init();
```

12.1.3 wl_apl_deinit

【原型】`int wl_apl_deinit (void)`

【功能】结束使用 APL 库，释放相关资源

【参数】无

【返回值】成功则返回 0，否则失败

【说明】同步函数，结束程序时调用

【举例】

```
//退出 APL
wl_apl_deinit();
```

12.1.4 wl_apl_point_down

【原型】`int wl_apl_point_down(int x, int y)`

【功能】单点触控，压下消息，具体位置要与 `wl_apl_set_config` 函数中设置 `wl_apl_screen_position_t` 所在的坐标系一致。

【参数】

x: x 坐标

y: y 坐标

【返回值】成功则返回 0，否则失败

【说明】同步函数

【举例】

```
//触控压下
wl_apl_point_down(100, 200);
```

12.1.5 wl_apl_point_up

【原型】`int wl_apl_point_up(int x, int y)`

【功能】单点触控，压下消息，具体位置要与 `wl_apl_set_config` 函数中设置 `wl_apl_screen_position_t` 所在的坐标系一致。

【参数】

x: x 坐标

y: y 坐标

【返回值】 成功则返回 0，否则失败

【说明】 同步函数

【举例】

//触控抬起

```
wl_apl_point_up(100, 200);
```

12.1.6 wl_apl_move

【原型】 `int wl_apl_point_move(int x, int y)`

【功能】 单点触控，压下消息，具体位置要与 wl_apl_set_config 函数中设置 wl_apl_screen_position_t 所在的坐标系一致。

【参数】

x: x 坐标

y: y 坐标

【返回值】 成功则返回 0，否则失败

【说明】 同步函数

【举例】

//触控移动

```
wl_apl_point_move(100, 200);
```

12.1.7 wl_apl_send_button

【原型】 `int wl_apl_send_button(int btn, int para)`

【功能】 发送按键信息

【参数】

key: 按键信息，主要指画面上可压下的 BUTTON，其中部分 BUTTON 可以控制手机端界面迁移，详见 [12.4.6](#)。

para: 备用参数，默认传递 0

【返回值】 成功则返回 0，否则失败

【说明】 同步函数

【举例】

//回到手机端 launcher 界面(WL_PAGE_BTN_LAUNCHER 压下)

```
wl_apl_send_button(WL_PAGE_BTN_LAUNCHER, 0);
```

12.1.8 wl_apl_send_msg

【原型】 `int wl_apl_send_msg(int msg, void *para)`

【功能】 发送消息到 APL

【参数】

msg: 消息类型, 详见 [12.4.4](#)

para: 消息参数, 如果不需要则传递 NULL

【返回值】成功则返回 0, 否则失败

【说明】同步函数

【举例】

//wmlink 程序切换到前台

```
wl_apl_send_msg(WL_APL_MSG_TYPE_APP_SHOW_WELINK, NULL);
```

//wmlink 程序切换到后台

```
wl_apl_send_msg(WL_APL_MSG_TYPE_APP_HIDE_WELINK, NULL);
```

//设置车机端蓝牙地址

```
wl_apl_send_msg(WL_APL_MSG_TYPE_APP_BT_ADDR, "3E:4C:2A:55:22:1D");
```

//设置车机端显示语言, 0 为中文, 1 为英文

```
int language = 0;
```

```
wl_apl_send_msg(WL_APL_MSG_TYPE_APP_SET_LANGUAGE, &language);
```

//设置行车状态, 0 停车, 1 行车。

```
int stat= 0;
```

```
wl_apl_send_msg(WL_APL_MSG_TYPE_APP_SET_DRIVE_STAT, &stat);
```

12.1.9 wl_apl_get_version

【原型】`const char *wl_apl_get_version(void)`

【功能】获取 APL 版本号

【参数】无

【返回值】成功则返回 APL 版本号, 失败返回 NULL

【说明】同步函数。

【举例】

//获取 APL 版本号

```
const char *version = wl_apl_get_version();
```

12.1.10 wl_apl_send_record_data

【原型】`int wl_apl_send_record_data(char *data, int size)`

【功能】将录音数据发送到手机（若选择车机端录音, 则调用此接口会将语音识别、微信回复等数据发送到手机端）

【参数】

data: 录音数据（16K、16bit、单声道）

size: 录音数据大小

【返回值】成功则返回发送字节长度, 否则失败

【说明】同步函数

【举例】

//发送 MIC 录音数据到手机端, 做语音识别用

```
wl_apl_send_record_data (data, size);
```

12.1.11 wl_apl_send_audio_data

【原型】`int wl_apl_send_audio_data(int type, char *data, int size)`

【功能】若选择 USB 传输手机端音频数据，可以通过此接口对手机端 PCM 数据协议进行回复，具体协议需要与手机端定义（参考：《WeLink 音频数据通信协议》）。

【参数】

type: 0（默认）

data: PCM 音频协议数据

size: 数据长度

【返回值】成功则返回发送字节长度，否则失败

【说明】同步函数

【举例】

//发送音频协议数据，具体协议需要与手机端定义

wl_apl_send_audio_data (0, data, size);

12.1.12 wl_apl_send_point

【原型】`int wl_apl_send_point(int x, int y, int id, int presure, wl_apl_touch_t type)`

【功能】多点触控函数，发送时序需要与手机端沟通，具体位置要与 wl_apl_set_config 函数中设置 wl_apl_screen_position_t 所在的坐标系一致。

【参数】

x: 横坐标

y: 纵坐标

id: 触控点 id，从 0 开始

presure: 压力，默认 50

type: 触控类型，详见 [12.4.8](#)

【返回值】成功则返回 0，否则失败

【说明】异步函数

【举例】

//第一个点压下

wl_apl_send_point (100, 100, 0, 50, WL_APL_TOUCH_LBUTTON_DOWN);

//第二个点压下

wl_apl_send_point (200, 200, 1, 50, WL_APL_TOUCH_LBUTTON_DOWN);

//一次触控完成

wl_apl_send_point_sync ();

//第一个点移动

wl_apl_send_point (110, 105, 0, 50, WL_APL_TOUCH_LBUTTON_MOVE);

//第二个点移动

wl_apl_send_point (205, 206, 1, 50, WL_APL_TOUCH_LBUTTON_MOVE);

//一次触控完成

wl_apl_send_point_sync ();

//第一个点抬起

wl_apl_send_point (110, 105, 0, 50, WL_APL_TOUCH_LBUTTON_UP);

```
//第二个点抬起
wl_apl_send_point (205, 206, 1, 50, WL_APL_TOUCH_LBUTTON_UP);
//一次触控完成
wl_apl_send_point_sync ();
```

12.1.13 wl_apl_send_point_sync

【原型】 `int wl_apl_send_point_sync(void)`

【功能】一次触控事件结束命令，具体发送时序需要与手机端沟通

【参数】

【返回值】成功则返回 0，否则失败

【说明】异步函数

【举例】

```
//第一个点压下
wl_apl_send_point (100, 100, 0, 50, WL_APL_TOUCH_LBUTTON_DOWN);
//一次触控完成
wl_apl_send_point_sync ();
//第一个点移动
wl_apl_send_point (110, 105, 0, 50, WL_APL_TOUCH_LBUTTON_MOVE);
//一次触控完成
wl_apl_send_point_sync ();
//第一个点抬起
wl_apl_send_point (110, 105, 0, 50, WL_APL_TOUCH_LBUTTON_UP);
//一次触控完成
wl_apl_send_point_sync ();
```

12.1.14 wl_apl_send_hard_key

【原型】 `int wl_apl_send_hard_key(int key, int para)`

【功能】物理按键命令，暂时不需关注

【参数】

key: 物理按键值，详见 [12.4.7](#)

para: 备用参数

【返回值】成功则返回 0，否则失败

【说明】同步函数

【举例】

```
//MIC 录音按键
wl_apl_send_hard_key (WL_API_HARD_KEY_RECORD, NULL);
```

12.1.15 wl_apl_device_chg

【原型】 `int wl_apl_device_chg (int device, char *ip, wl_apl_device_chg_info_t*deviceinfo)`

【功能】设备状态变化

【参数】

device: 设备类型, 取值详见 [12.4.9](#)

ip: 设备 ip 地址, 仅无线需要, 如果是 USB 连接则传递 NULL

deviceinfo: 设备信息, 仅 USB 连接需要, 如果是无线连接则传 NULL, 详见 [12.3.10](#)

【返回值】成功则返回 0, 否则失败

【说明】同步函数

【举例】

//苹果无线设备插入

```
wl_apl_send_hard_key (WL_APL_DEVICE_WLAN_IPHONE, "192.168.2.124", NULL);
```

12.1.16 wl_apl_get_wlan_list

【原型】`wl_apl_wlan_node_t * wl_apl_get_wlan_list (void)`

【功能】获取无线设备列表, 设备列表信息参照 [12.3.8](#)

【参数】

【返回值】返回无线设备节点指针, 失败返回 NULL

【说明】同步函数

【举例】

//获取无线设备列表

```
wl_apl_wlan_node_t * wlan_node = wl_apl_get_wlan_list();
```

12.1.17 wl_apl_send_hu_data

【原型】`int wl_apl_send_hu_data(int type, char *data, int size)`

【功能】将车身数据消息发送到手机, 具体协议需要与手机端定义, 暂时不需关注

【参数】

type: 0 (默认)

data: 车身消息数据

size: 数据长度

【返回值】成功则返回发送字节长度, 否则失败

【说明】同步函数

【举例】

//发送车身协议数据, 具体协议需要与手机端定义

```
wl_apl_send_hu_data (0, data, size);
```

12.2 回调函数

APL 通过回调函数与上层 APP 通信，下面进行详细说明。

12.2.1 update_page

【原型】`typedef int (*update_page)(int page, void *para)`

【功能】APL 调用此函数，通知 APP 页面更新

【参数】

page: 页面信息，详见 [9.1](#)

para: 备用参数，具体使用可参照下面示例代码

【返回值】成功则返回 0，否则失败

【说明】此函数是同步函数，可以在此函数中进行描画操作，对 para 的操作需要在函数结束前进行，函数返回后，para 指针不能再继续使用。

【举例】

```
static int _p_update_page(int type, void *para)
{
    switch (type) {
        case WL_PAGE_SOURCE_OFF: //切换到后台，不需要描画
            //此处追加画面隐藏处理
            break;
        case WL_ERR_PAGE_AUTH:
            if (NULL != para) {
                const char * err_code = (char *)para; //错误码
                //此处描画错误码显示画面
            }
            break;
        case WL_RUNNING_PAGE_WECHAT_QRCODE:
            if (NULL != para) {
                wl_apl_rgb_info_t *info = (wl_apl_rgb_info_t *)para; //微信登陆二维码数据
                //此处描画微信登陆二维码界面
            }
            break;
        case WL_RUNNING_PAGE:
            //此处追加投影手机端数据前的准备处理
            break;
        default:
            //其余画面不需要关注 para 参数，具体参照 9.1
            break;
    }
    return 0;
}
```

12.2.2 recv_msg

【原型】`typedef int (*recv_msg)(int msg, void *para)`

【功能】APL 调用此函数，与 APP 进行消息通信

【参数】

msg: 消息类型，详见 [12.4.4](#)

para: 备用参数，默认为 NULL，具体使用可参照示例代码

【返回值】成功则返回 0，否则失败

【说明】此函数是同步函数，对 para 的操作需要在函数结束前进行，函数返回后，para 指针不能再继续使用

【举例】

```
static int _p_recv_msg(int msg, void *para)
{
    switch (msg) {
        case WL_APL_MSG_TYPE_APL_REQ_SHOW: //请求显示 WeLink
            //显示 WeLink
            wl_apl_send_msg(WL_APL_MSG_TYPE_APP_SHOW_WELINK, NULL);
            break;
        case WL_APL_MSG_TYPE_APL_REQ_HIDE: //请求隐藏 WeLink
            //隐藏 WeLink
            wl_apl_send_msg(WL_APL_MSG_TYPE_APP_HIDE_WELINK, NULL);
            break;
        case WL_APL_MSG_TYPE_APL_OPEN_BT: //请求打开蓝牙
            //此处添加打开蓝牙处理， 如果不需要自动连接蓝牙则可以不关注
            break;
        case WL_APL_MSG_TYPE_APL_BT_STATUS: //蓝牙状态通知
            if (NULL != para) {
                int type = *(int*)para; //0 为未连接， 1 为已连接
                //此处根据需求决定是否切换到蓝牙连接画面， 由用户手动连接蓝牙
            }
            break;
        case WL_APL_MSG_TYPE_APL_REQ_BT_ADDR:
            //通知 APL 车机端蓝牙地址
            wl_apl_send_msg(WL_APL_MSG_TYPE_APP_BT_ADDR, "3E:4C:2A:55:22:1D");
            break;
        case WL_APL_MSG_TYPE_APL_RECV_MSG:
            if (NULL != para) {
                int type = *(int*)para; //消息类型， 1 为微信， 3 为其他
                //此处添加收到消息图像闪烁处理
            }
            break;
        case WL_APL_MSG_TYPE_APL_NAVI_GUIDE_START:
        case WL_APL_MSG_TYPE_APL_NAVI_GUIDE_UPDATE:
```



```

    if (NULL != para) {
        wl_apl_navi_guide_t info = *(wl_apl_navi_guide_t*)para;
        //此处添加显示导航条信息处理
    }
    break;
case WL_APL_MSG_TYPE_APL_NAVI_GUIDE_STOP:
    //此处添加停止显示导航条信息处理
    break;
case WL_APL_MSG_TYPE_APL_UPDATE_KEY:
    if (NULL != para) {
        //当前高亮的 KEY，特指常住条下的 button，取值从
        //WL_PAGE_BTN_LAUNCHER 到 WL_PAGE_BTN_MIC，如果 key 值为 0，
        //都不高亮显示。
        int key = *(int *)para;
        //此处添加 Button 高亮显示处理
    }
    break;
case WL_APL_MSG_TYPE_APL_CONNECTING: //连接中状态通知
    break;
case WL_APL_MSG_TYPE_APL_CONNECTED: //连接成功状态通知
    break;
case WL_APL_MSG_TYPE_APL_CONNECT_ERR: //连接失败状态通知
    break;
case WL_APL_MSG_TYPE_APL_DISCONNECT: //断开连接
    break;
case WL_APL_MSG_TYPE_APL_START_RECORD: //开始录音
    //此处添加启动录音线程处理，并在录音线程中将录音数据发送到手机
    //通过函数 wl_apl_send_record_data(3.1.12)
    break;
case WL_APL_MSG_TYPE_APL_STOP_RECORD: //停止录音
    //此处添加结束录音线程处理
    break;
case WL_APL_MSG_TYPE_APL_WLAN_LIST_UPDATE: //无线列表更新
{
    //获取无线设备列表
    wl_apl_wlan_node_t * node = wl_apl_get_wlan_list ();
    if (NULL != node && NULL != node->ip) {
        //取列表中第一个设备，通知 APL，无线设备 dev_info 默认传 NULL
        wl_apl_device_chg(node-> device_type, node->ip, NULL);
    }
}
    break;
default:
    //其他消息具体参照 12.4.4

```

```
        break;
    }
```

12.2.3 audio_data

【原型】 `typedef int (*audio_data)(void *data, int len)`

【功能】 APL 返回手机端音频数据到上层，如果车机不支持 PCM 数据播放则不需关注

【参数】

data: 手机端音频数据包，具体格式需要与手机端自定义

len: 音频数据包长度

【返回值】 成功则返回 0，否则失败

【说明】 此函数是同步函数，对 data 的操作需要在函数结束前进行，函数返回后，data 指针不能再继续使用。

【举例】

```
static int _p_audio_data(void *data, int len)
{
    //此处添加 audio 数据包解析及播放处理
    return 0;
}
```

12.2.4 hu_data

【原型】 `typedef int (*hu_data)(void *data, int len, int type)`

【功能】 APL 返回手机端与车机定义的数据，具体需要与手机端定义，暂时不需关注

【参数】

data: 通信数据内容

len: 数据长度

type: 默认 0

【返回值】 成功则返回 0，否则失败

【说明】 此函数是同步函数，对 data 的操作需要在函数结束前进行，函数返回后，data 指针不能再继续使用。

【举例】

```
static int _p_hu_data(void *data, int len, int type)
{
    //此处添加车身数据包解析处理
    return 0;
}
```

12.2.5 video_data

【原型】 `typedef int (*video_data)(wl_apl_image_info_t *data)`

【功能】APL 返回手机端图像数据

【参数】

data: 图像数据内容, 参照 [12.3.4](#)

【返回值】成功则返回 0, 否则失败

【说明】此函数是同步函数, 对 data 的操作需要在函数结束前进行, 函数返回后, data 指针不能再继续使用。

【举例】

```
static int _p_video_data(wl_apl_image_info_t *data)
{
    if (NULL != data) {
        wl_apl_image_info_t *info = (wl_apl_image_info_t *)para; //手机端图像数据
        //此处描画手机图像显示界面
    }
    return 0;
}
```

12.3 数据结构

12.3.1 APL 回调函数

表 12-2 对回调函数结构进行了详细说明。

表 12-2 数据结构 wl_apl_cb_t

名称	类型	说明	备注
func_update_page	update_page	页面更新回调函数	无
func_msg_notice	recv_msg	消息回调函数	无
func_audio_data	audio_data	Audio 数据包回调函数	无
func_hu_data	hu_data	保留 (reserve)	无
func_video_data	video_data	图像数据回调函数	无

12.3.2 RGB 图像信息

RGB 图像相关信息, 具体内容见表 12-3。

表 12-3 数据结构 wl_apl_rgb_info_t

名称	类型	说明	备注
width	int	图像宽度	无
height	int	图像高度	无
size	int	图像数据长度	无
data	void *	图像数据指针	无
format	wl_apl_image_format_t	图像数据格式	详见 12.4.2

12.3.3 H.264 数据信息

H.264 图像相关信息，具体内容见表 12-4。

表 12-4 数据结构 wl_apl_h264_info_t

名称	类型	说明	备注
spsppsDataSize	int	SPS 和 PPS 数据的长度	无
width	int	图像数据宽	无
height	int	图像数据高	无
size	int	图像数据长度	无
data	void*	图像数据指针	无
frameType	wl_apl_h264_fream_type_t	图像帧类型	详见 12.4.1

12.3.4 图像数据信息

手机端图像数据相关信息，具体内容见表 12-5

表 12-5 数据结构 wl_apl_image_info_t

名称	类型	说明	备注
type	wl_apl_image_type_t	图像数据类型	详见 12.4.3
rgb_info	wl_apl_rgb_info_t	RGB 图像信息	详见 12.3.2
h264_info	wl_apl_h264_info_t	H.264 图像信息	详见 12.3.3

12.3.5 屏幕相关信息

手机端图像在 UI 画面上输出的相关位置参数，具体内容见表 12-6。

表 12-6 数据结构 wl_apl_screen_position_t

名称	类型	说明	备注
x	int	手机端图像显示位置起始横坐标	无
y	int	手机端图像显示位置起始纵坐标	无
w	int	手机端图像显示位置宽度	无
h	int	手机端图像显示位置高度	无

12.3.6 APL 配置信息

APL 配置相关信息，具体内容见表 12-7。

表 12-7 数据结构 wl_apl_config_t

名称	类型	说明	备注
config_file_path	char *	配置文件 wl_info.ini 存放的路径，设置为空则默认获取当前路径。	无
huVersion	char *	车机端版本号	无

serialNum	char *	授权码	无
cb_func	wl_apl_cb_t	APL 回调函数	详见 12.3.1
fmt	wl_apl_image_format_t	RGB 数据格式	详见 12.3.2
screen	wl_apl_screen_position_t	手机端图像显示位置信息	详见 12.3.5
ssl_file	wl_apl_ssl_file_t	加密所用证书等信息	详见 12.3.9








12.3.7 导航条信息

导航条相关信息，具体内容见表 12-8。


表 12-8 数据结构 wl_apl_navi_guide_t

名称	类型	说明	备注
icon	int	转向标索引 字段说明详见表 12-9	无
limitSpeed	int	限速，单位：km/h	无
percentToCurrPoint	int	拐点进度条百分比，满格为 100	无
name	char*	当前道路名称	无
nextName	char*	下一条道路名称	无
direction	char*	地图朝向	无
remainDistance	char*	当前位置到终点的距离	无
remainTime	char*	当前位置到终点所需的预估时间	无
currSpeed	char*	当前时速，单位：km/h	无

表 12-9 转向标索引说明

编号	说明	图标示例	备注
0	起点		无
1	终点		无
2	调头		无
3,7,10	左前		无
4,6,11	右前		无
5,39,40	直行		无
8	左转		无

9	右转		无
12,13,14,15,16,17,18	环岛		无
21	保持左侧行驶		无
22	保持右侧行驶		无
23	向左急转弯		无
24	向右急转弯		无
25	向左转并保持左侧行驶		无
26	向左转并保持右侧行驶		无
27	向右转并保持左侧行驶		无
28	向右转并保持右侧行驶		无
29	进入隧道		无
37,38	非盘桥立交桥&盘桥立交桥		无
49	三岔路左转		无
50	三岔路直行		无

51	三岔路右转		无
----	-------	--	---

12.3.8 无线设备节点

无线设备节点相关信息，具体内容见表 12-10。

表 12-10 数据结构 wl_apl_wlan_node_t

名称	类型	说明	备注
device_type	int	设备类型，取值详见 12.4.9	无
name	char *	设备名	无
ip	char *	IP 地址	无
mac	char *	MAC 地址	无
next	wl_apl_wlan_node_t *	下一节点	无

12.3.9 加密相关信息

加密使用的相关信息，具体内容见表 12-11。

表 12-11 数据结构 wl_apl_ssl_file_t

名称	类型	说明	备注
ca_file	char *	CA 证书内容	无
cert_file	char *	用户证书	无
cert_pass	char *	用户私钥的口令，保护私钥文件	无
cert_key	char *	用户私钥	无

12.3.10 USB 设备信息

USB 设备相关信息，wl_apl_device_chg ([12.1.15](#)) 函数使用参数，无线方案不需关注，具体内容见表 12-12。

表 12-12 数据结构 wl_apl_device_chg_info_t

名称	类型	说明	备注
vid	int	设备 VID，默认 0xFF	无
pid	int	设备 PID，默认 0xFF	无
protocol	int	私有协议，默认 0xFF	无
usb_info	char *	USB 信息，不同车机不同设置，默认 NULL	无

12.4 枚举

本节对 APL 中使用到的枚举结构进行详细说明。

12.4.1 H.264 图像类型

H.264 图像帧类型具体定义见表 12-13。

表 12-13 枚举结构 wl_apl_h264_fream_type_t

名称	值	说明	备注
WL_APL_H264_FREAM_TYPE_SPS_IDR	1	带 SPS 和 PPS 信息的 IDR 帧	无
WL_APL_H264_FREAM_TYPE_IDR	2	IDR 帧	无
WL_APL_H264_FREAM_TYPE_P	3	P 帧	无
WL_APL_H264_FREAM_TYPE_ERR	4	错误帧类型	无

12.4.2 RGB 图像格式

RGB 图像格式具体定义见表 12-14，其中部分格式暂不支持。

表 12-14 枚举结构 wl_apl_image_format_t

名称	值	说明	备注
WL_APL_IMAGE_FORMAT_RGB	0	RGB 格式	支持
WL_APL_IMAGE_FORMAT_BGR	1	BGR 格式	支持
WL_APL_IMAGE_FORMAT_RGBX	2	RGBX 格式	支持
WL_APL_IMAGE_FORMAT_BGRX	3	BGRX 格式	支持
WL_APL_IMAGE_FORMAT_XBGR	4	XBGR 格式	支持
WL_APL_IMAGE_FORMAT_XRGB	5	XRGB 格式	支持
WL_APL_IMAGE_FORMAT_GRAY	6	GRAY 格式	支持
WL_APL_IMAGE_FORMAT_RGBA	7	RGBA 格式	支持
WL_APL_IMAGE_FORMAT_BGRA	8	BGRA 格式	支持
WL_APL_IMAGE_FORMAT_ABGR	9	ABGR 格式	支持
WL_APL_IMAGE_FORMAT_ARGB	10	ARGB 格式	支持
WL_APL_IMAGE_FORMAT_CMYK	11	CMYK 格式	暂不支持

12.4.3 图像数据格式定义

APL 向上层传递数据时，会告知图像格式，具体内容在表 12-15 中定义。

表 12-154 枚举结构 wl_apl_image_type_t

名称	值	说明	备注
WL_APL_IMAGE_TYPE_RGB	0	RGB 数据	无
WL_APL_IMAGE_TYPE_H264	1	H264 原始数据	无
WL_APL_IMAGE_TYPE_JPG	2	JPG 原始数据	无
WL_APL_IMAGE_TYPE_PNG	3	PNG 原始数据	无
WL_APL_IMAGE_TYPE_BMP	4	BMP 原始数据	无

12.4.4 通信消息类型定义

APL 向上层传递消息时，消息类型从表 12-16 中取值，其中部分消息需要根据项目定制。

表 12-16 枚举结构 wl_apl_msg_type_t

名称	值	功能	说明	备注
WL_APL_MSG_TYPE_APL_REQ_SHOW	1	请求显示	共通	APL->APP
WL_APL_MSG_TYPE_APL_REQ_HIDE	2	请求隐藏	共通	APL->APP
WL_APL_MSG_TYPE_APL_OPEN_BT	3	请求打开蓝牙	项目定制	APL->APP
WL_APL_MSG_TYPE_APL_BT_STATUS	4	手机端蓝牙状态，参数传 int*，1 为已连接，0 为未连接	项目定制	APL->APP
WL_APL_MSG_TYPE_APL_REQ_BT_ADDR	5	请求蓝牙地址	共通	APL->APP
WL_APL_MSG_TYPE_APL_RECV_MSG	6	收到信息，此时需要根据参数，闪烁消息图标	共通	APL->APP
WL_APL_MSG_TYPE_APL_NAVI_GUIDE_START	7	开始显示导航条信息	项目定制	APL->APP
WL_APL_MSG_TYPE_APL_NAVI_GUIDE_UPDATE	8	更新导航条信息	项目定制	APL->APP
WL_APL_MSG_TYPE_APL_NAVI_GUIDE_STOP	9	停止显示导航条信息	项目定制	APL->APP
WL_APL_MSG_TYPE_APL_UPDATE_KEY	10	当前高亮的 KEY，特指常住条上的 KEY，参数传 int*，取值从 WL_PAGE_BTN_LAUNCHER 到 WL_PAGE_BTN_MIC，如果 KEY 值为 0，则都不高亮显示	共通	APL->APP
WL_APL_MSG_TYPE_APL_CONNECTING	11	连接中	共通	APL->APP
WL_APL_MSG_TYPE_APL_CONNECTED	12	连接成功	共通	APL->APP
WL_APL_MSG_TYPE_APL_CONNECT_ERR	13	连接失败	共通	APL->APP
WL_APL_MSG_TYPE_APL_DISCONNECT	14	断开连接	共通	APL->APP
WL_APL_MSG_TYPE_APL_START_RECORD	15	开始录音	项目定制	APL->APP
WL_APL_MSG_TYPE_APL_STOP_RECORD	16	结束录音	项目定制	APL->APP
WL_APL_MSG_TYPE_APL_WLAN_LIST_UPDATE	17	无线设备列表更新	项目定制	APL->APP
WL_APL_MSG_TYPE_APP_SHOW_WELINK	18	显示	共通	APP->APL
WL_APL_MSG_TYPE_APP_HIDE_WELINK	19	隐藏	共通	APP->APL
WL_APL_MSG_TYPE_APP_BT_ADDR	20	蓝牙地址，参数传 char*，例如：“2C:4D:33:55:10:2A”	共通	APP->APL
WL_APL_MSG_TYPE_APP_SET_LANGUAGE	21	设定显示语言，参数传 int*，0 中文，1 英文	项目定制	APP->APL
WL_APL_MSG_TYPE_APP_SET_DRIVE_STAT	22	设定行驶车状态，参数传 int*，0 停车，1 行车	项目定制	APP->APL
WL_APL_MSG_TYPE_MAX_NUM	23	不需关注	共通	无

12.4.5 页面类型定义

页面类型是 APL 通过回调函数 `update_page` ([12.2.1](#)) 通知 APP，之后 APP 再根据不同的页面类型来描画相关的页面，详细参照表 12-17。

表 12-17 枚举结构 `wl_apl_page_type_t`

名称	值	说明	备注
WL_PAGE_SOURCE_OFF	0	welink 切换到后台，停止描画	无
WL_INITIAL_PAGE	1	初始化界面	无
WL_INITIAL_PAGE_NOT_AUTH	2	未打开 USB 调试界面	仅 ADB 方案
WL_HELP_PAGE_ANDROID	3	帮助界面（Android 手机）	无
WL_HELP_PAGE_IOS	4	帮助界面（Iphone 手机）	无
WL_CONNECTING_PAGE_OFFICIAL_ACCOUNT	5	WeLink 公众号画面	无
WL_CONNECTING_PAGE	6	连接中画面	无
WL_CONNECTING_PAGE_INSTALL	7	安装中画面	仅 ADB 方案
WL_CONNECTING_PAGE_AUTH	8	权限验证中画面	无
WL_ERR_PAGE_SERVICE	9	手机端服务启动失败	仅 ADB 方案
WL_ERR_PAGE_SDK	10	手机端版本过低	仅 ADB/AOA 方案
WL_ERR_PAGE_CONNECT_ADB	11	ADB 初次连接失败画面	仅 ADB 方案
WL_ERR_PAGE_CONNECT	12	连接失败	无
WL_ERR_PAGE_INSTALL	13	安装失败	仅 ADB 方案
WL_ERR_PAGE_AUTH	14	认证失败，错误码根据 <code>para</code> 参数确定（ <code>char*</code> ）	无
WL_ERR_PAGE_AUTH_OVERDUE	15	认证失败（授权过期）	无
WL_ERR_PAGE_AUTH_NO_NETWORK	16	认证失败（手机无网络）	无
WL_ERR_PAGE_HEART	17	心跳错误	无
WL_RUNNING_PAGE_SAFE_DRIVING	18	安全驾驶界面，行车状态下会禁止用户部分操作	无
WL_RUNNING_PAGE_WECHAT_QRCODE	19	微信二维码登录界面，二维码数据根据参数 <code>para</code> 确认（ <code>wl_apl_rgb_info_t*</code> ）	无
WL_RUNNING_PAGE_WECHAT_CONFIRM_LOGIN	20	微信登录确认界面	无
WL_RUNNING_PAGE_WECHAT_LOGIN_SUCCESS	21	微信登录成功界面	无
WL_RUNNING_PAGE_SCREEN_OFF	22	锁屏画面	无
WL_RUNNING_PAGE_LIMIT	23	限制投影画面	无
WL_RUNNING_PAGE_WAIT_FOR_VIDEO	24	等待手机端图像数据界面，暂时不需要处理	不需要关注
WL_RUNNING_PAGE	25	手机图像投影画面，手机端图像数据根据参数 <code>para</code> 确认（ <code>wl_apl_image_info_t*</code> ）	无
WL_PAGE_NUM	26	不需关注	无

12.4.6 页面按键类型定义

页面按键类型是 APP 通过函数传递给 APL，在对应的按键出现压下抬起动作后，APP 调用 `wl_apl_send_button` 函数（详见 [12.1.7](#)）来通知 APL，按键信息具体参照表 12-18。

表 12-18 枚举结构 `wl_apl_button_t`

名称	值	说明	备注
WL_PAGE_BTN_HOME	1	常驻条按键，返回车机主界面	无
WL_PAGE_BTN_LAUNCHER	2	常驻条按键，切换到手机 launcher 界面	可以控制手机端界面迁移
WL_PAGE_BTN_NAVI	3	常驻条按键，切换到手机导航界面	可以控制手机端界面迁移
WL_PAGE_BTN_PHONE	4	常驻条按键，切换到手机电话界面	可以控制手机端界面迁移
WL_PAGE_BTN_MUSIC	5	常驻条按键，切换到手机音乐界面	可以控制手机端界面迁移
WL_PAGE_BTN_MSG	6	常驻条按键，切换到手机消息界面	可以控制手机端界面迁移
WL_PAGE_BTN_MORE	7	常驻条按键，切换到手机更多界面	可以控制手机端界面迁移
WL_PAGE_BTN_MIC	8	常驻条按键，切换到手机录音界面	可以控制手机端界面迁移
WL_PAGE_BTN_NONE_EXIT	9	初始化界面退出按键（返回车机主界面）	无
WL_PAGE_BTN_HELP_ANDROID	10	Android 手机帮助按键	无
WL_PAGE_BTN_HELP_IOS	11	苹果手机帮助按键	无
WL_PAGE_BTN_HELP_BACK	12	帮助界面返回按键	无
WL_PAGE_BTN_SAFETY_CLOSE	13	关闭行车安全界面	无
WL_PAGE_BTN_CLICK_HERE	14	“请点击这里”按键	无
WL_PAGE_BTN_RETRY	15	重试连接按键	无
WL_PAGE_BTN_GOT_IT	16	“知道了”按键	无
WL_PAGE_BTN_ERR_CLOSE	17	错误画面退出按键（返回车机主界面）	无
WL_PAGE_BTN_LIMIT_CLOSE	18	限制投影画面退出按键（返回车机主界面）	无
WL_PAGE_BTN_NAVI_GUIDE_CLOSE	19	关闭导航条信息界面	无
WL_PAGE_BTN_WECHAT_QRCODE_CLOSE	20	微信登录二维码界面退出按键	无
WL_PAGE_BTN_WECHAT_CANCEL_LOGIN	21	微信登录界面退出按键	无
WL_PAGE_BTN_NOT_AUTH_OK	22	未打开 USB 调试画面下的 OK 按键	仅 ADB 方案
WL_PAGE_BTN_NOT_AUTH_CANCEL	23	未打开 USB 调试画面下的退出按键	仅 ADB 方案
WL_PAGE_BTN_INSTALL_ERR_CLOSE	24	安装失败画面退出按键	仅 ADB 方案

WL_PAGE_BTN_OFFICIAL_ACCOUNT_SKIP	25	WeLink 公众号页面“跳过”按钮	无
WL_PAGE_BTN_MAX_NUM	26	不需关注	无

12.4.7 物理按键

物理按键类型具体定义见表 12-19，功能暂未实现，无需关注。

表 12-19 枚举结构 wl_apl_hard_key_t

名称	值	说明	备注
WL_API_HARD_KEY_RECORD	1	MIC 录音键	无
WL_API_HARD_KEY_PRE	2	上一曲	无
WL_API_HARD_KEY_NEXT	3	下一曲	无
WL_API_HARD_KEY_NUM	4	无需关注	无

12.4.8 触控类型

触控类型具体定义见表 12-20。

表 12-20 枚举结构 wl_apl_touch_t

名称	值	说明	备注
WL_APL_TOUCH_LBUTTON_UNKNOWN	0	不需关注	无
WL_APL_TOUCH_LBUTTON_DOWN	1	压下	无
WL_APL_TOUCH_LBUTTON_UP	2	抬起	无
WL_APL_TOUCH_LBUTTON_MOVE	3	移动	无

12.4.9 设备类型

设备类型具体定义见表 12-21。

表 12-21 设备类型

名称	值	说明	备注
WL_APL_DEVICE_OUT	1	设备拔出	无
WL_APL_DEVICE_AOA	7	安卓 AOA 设备	无
WL_APL_DEVICE_WLAN_IPHONE	9	苹果无线设备	无
WL_APL_DEVICE_WLAN_ANDROID	10	安卓无线设备	无
WL_APL_DEVICE_EAP	11	苹果 EAP 设备	无

13. APL 的具体使用

13.1 APL 文件说明

APL 会提供相关的库文件、头文件以及运行需要的相关文件，APP 将 APL 编译到程序中，生成可执行程序，并和运行需要的相关文件放到一起即可正常运行，表 13-1 对 APL 提供的文件进行具体说明，针对不同的需求，会提供不同的文件。

表 13-1 文件说明

No	名称	说明	备注
1	wl_api.h	APL 头文件	程序编译使用文件
2	libwl_apl_shared.so	APL 库文件	程序编译、运行使用文件
3	wl_info.ini	应用程序配置文件	程序运行使用文件
4	wlDriver	驱动文件(AOA/EAP 方案使用)	程序运行使用文件 需要 root 权限启动
5	welinkDevice	驱动文件(ADB 方案使用)	程序运行使用文件 需要 root 权限启动
6	welink	文件夹，手机端服务文件(ADB 方案使用)	程序运行使用文件
7	WeDrive.apk	手机端应用程序(ADB 方案使用)	程序运行使用文件
8	WeDrive_AppStore.apk	手机端应用程序(ADB 方案使用)	程序运行使用文件

其中，No.2-No.8 文件，默认和使用 libwl_apl_shared.so 编译生成的运行程序，放在同级目录下；

配置文件 wl_info.ini 主要用于动态配置 APL，由 APL 提供，APP 不可修改。

13.2 APL 使用步骤

13.2.1 配置并初始化 WeLink APL

在与 WeLink APL 进行数据通信前，需要首先对 APL 进行配置并初始化。

使用以下函数完成对 APL 的配置：

```
int wl_apl_set_config(wl_apl_config_t *config);
```

使用以下函数进行 APL 初始化：

```
int wl_apl_init(void);
```

参数 config 结构可以参照 [12.1.1](#) 中相应说明对每项成员进行设定。

13.2.2 手机设备接入与断开

如前文描述，AOA、EAP、WIFI 方式互联，需要车机端 APP 来通知 WeLink 设备接入与断开，那么可以调用以下函数来实现：

```
int wl_apl_device_chg(int device, char* ip, char* deviceinfo);
```

另外，手机通过 WIFI 方式接入的情况，简单来说有以下几步：

1. WeLink 获取已经与车机处于同一网段（已经打开趣驾 WeLink APP）的手机设备列表；
2. WeLink 向上层发送“WL_APL_MSG_TYPE_APL_WLAN_LIST_UPDATE”消息，通知上层应用可以获取该手机列表；
3. 上层应用接收到该消息后，调用以下接口函数来取得接入的手机设备列表：

```
wl_apl_wlan_node_t* wl_apl_get_wlan_list(void);
```

4. 上层应用选择一个即将接入的设备，将该设备信息通过接口 `wl_apl_device_chg` 通知 WeLink，WeLink 开始互联。

Note: 与车机处于同一网段的手机设备列表发生变化（如有手机“加入”或“离开”车机网段），WeLink 都会向上发送“WL_APL_MSG_TYPE_APL_WLAN_LIST_UPDATE”消息。

13.2.3 处理 APL 返回的消息

WeLink APL 的状态主要通过消息不断地（回调）反馈给车机端应用，车机端应用在初始化完成后，就需要实时对 APL 的状态进行监控管理并进行相应的处理，参照 [12.2.2](#)。

13.2.4 向 APL 发送消息

车机端应用使用如下接口函数向 WeLink APL 发送消息，参照 [12.1.8](#)。

上层应用需要关注的消息如下：

WL_APL_MSG_TYPE_APP_SHOW_WELINK: 通知 APL 显示 Welink

WL_APL_MSG_TYPE_APP_HIDE_WELINK: 通知 APL 隐藏 Welink

WL_APL_MSG_TYPE_APP_BT_ADDR: 向 APL 发送蓝牙地址

13.2.5 页面更新处理

除了前面描述的状态消息之外，WeLink 在各个场景下需要描画的页面，也是由 APL 向上层应用反馈的（同样是回调方式）。车机端应用开发者在接收到页面更新消息后，进行相应的页面描画即可，参照 [12.2.1](#)。

13.2.6 音频数据的播放

WeLink“出声”时，会将 pcm 数据通过回调的方式，传递给车机端应用，回调函数参照 [12.2.3](#)。

13.2.7 语音识别

关于 WeLink 的语音助手功能，车机端在接收到 WeLink “WL_APL_MSG_TYPE_APL_START_RECORD” 的消息后，需要实时地采集 Mic 数据传送给 WeLink，对应的接口参照 [12.1.10](#)。在接收到 WeLink “WL_APL_MSG_TYPE_APL_STOP_RECORD” 的消息后，停止传递 Mic 数据。

13.2.8 回控的使用

为了能在车机屏幕上点击触控使用 WeLink，上层应用需要将触控坐标点发送给 WeLink，参照 [第七章](#)。

13.2.9 其他

其他功能根据各自需求，直接参考 Sample code 和上文 APL 说明即可，这里不一一赘述。

14. FAQ

T. B. D

WeLink