

RK3308 RTL8723DS 蓝牙笔记

(技术部，系统产品一部)

| | | |
|-------------------------------|-------|------------|
| 文件状态： [√] 正在修改 [] 正式发布 | 当前版本： | V1.0 |
| | 作 者： | 许学辉 |
| | 完成日期： | 2018-11-10 |
| | 审 核： | |
| | 完成日期： | |

福州瑞芯微电子有限公司

Fuzhou Rockchips Semiconductor Co., Ltd

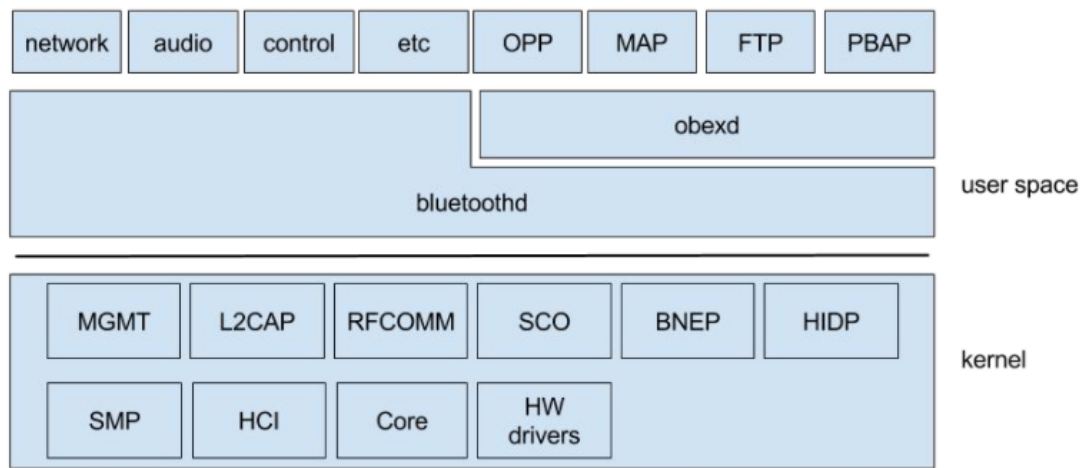
(版本所有, 翻版必究)

1 目的

记录一下基于 BLUEZ 的 8723DS 蓝牙相关开发笔记，调试记录和对协议栈的一些理解。目前来看 LINUX 平台上很多东西不是 RK 定标准，RK 经验也不足，对协议栈的理解是基于项目和平时对代码的理解进行总结，有的地方可能不对。

很多逻辑和流程都是客户那边进行定义，在逻辑和控制上难免有各种问题，我们的处理方式也有些欠妥的地方，对于底层的控制，由于客户不负责，比如联网，BLE 广播，蓝牙控制，蓝牙逻辑处理，A2DP 问题分析，蓝牙 FW 相关问题只能 RK 配合来解决

BlueZ 是官方 Linux Bluetooth 栈，由主机控制接口（Host Control Interface ， HCI）层、Bluetooth 协议核心、逻辑链路控制和适配协议（Logical Link Control and Adaptation Protocol， L2CAP）、SCO 音频层、其他 Bluetooth 服务、用户空间后台进程以及配置工具组成。BlueZ 提供对核心蓝牙层和协议的支持。**bluez 的特点网上找一大堆。我们一般关心 user space 即可，内核一般改不动，如下截图来着网上，网上一大堆**



Bluetooth core system architecture

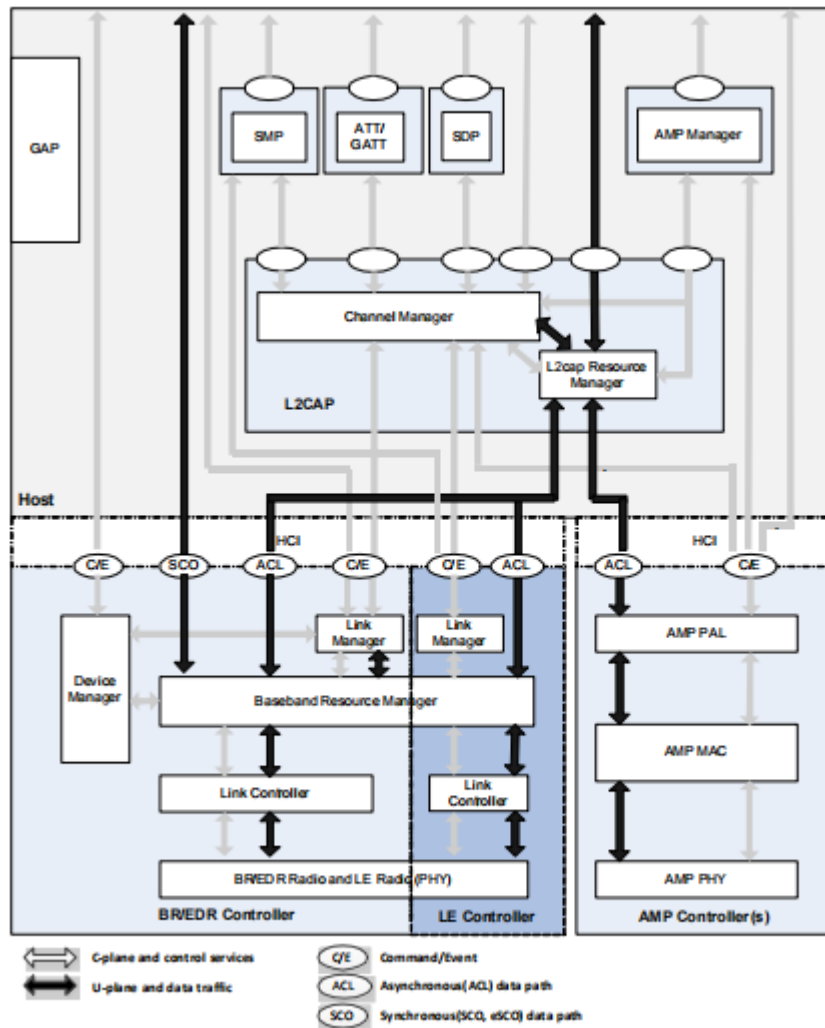
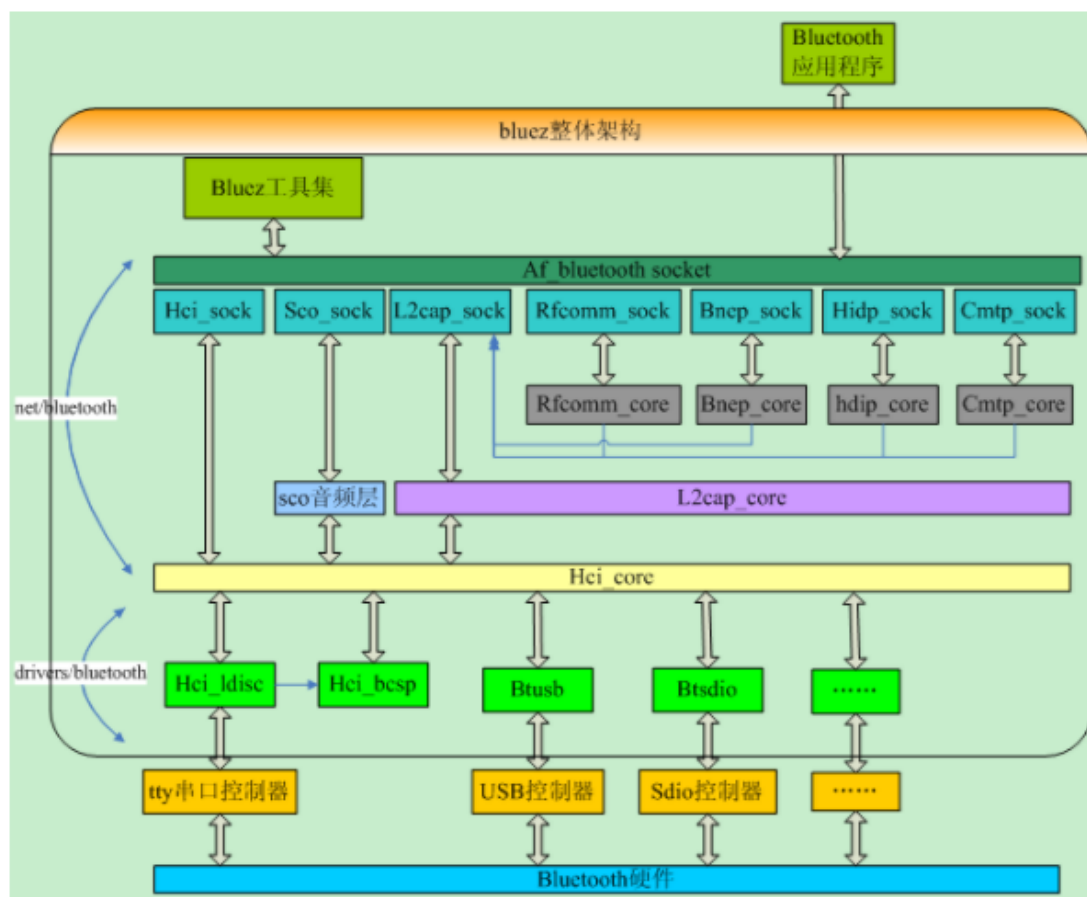


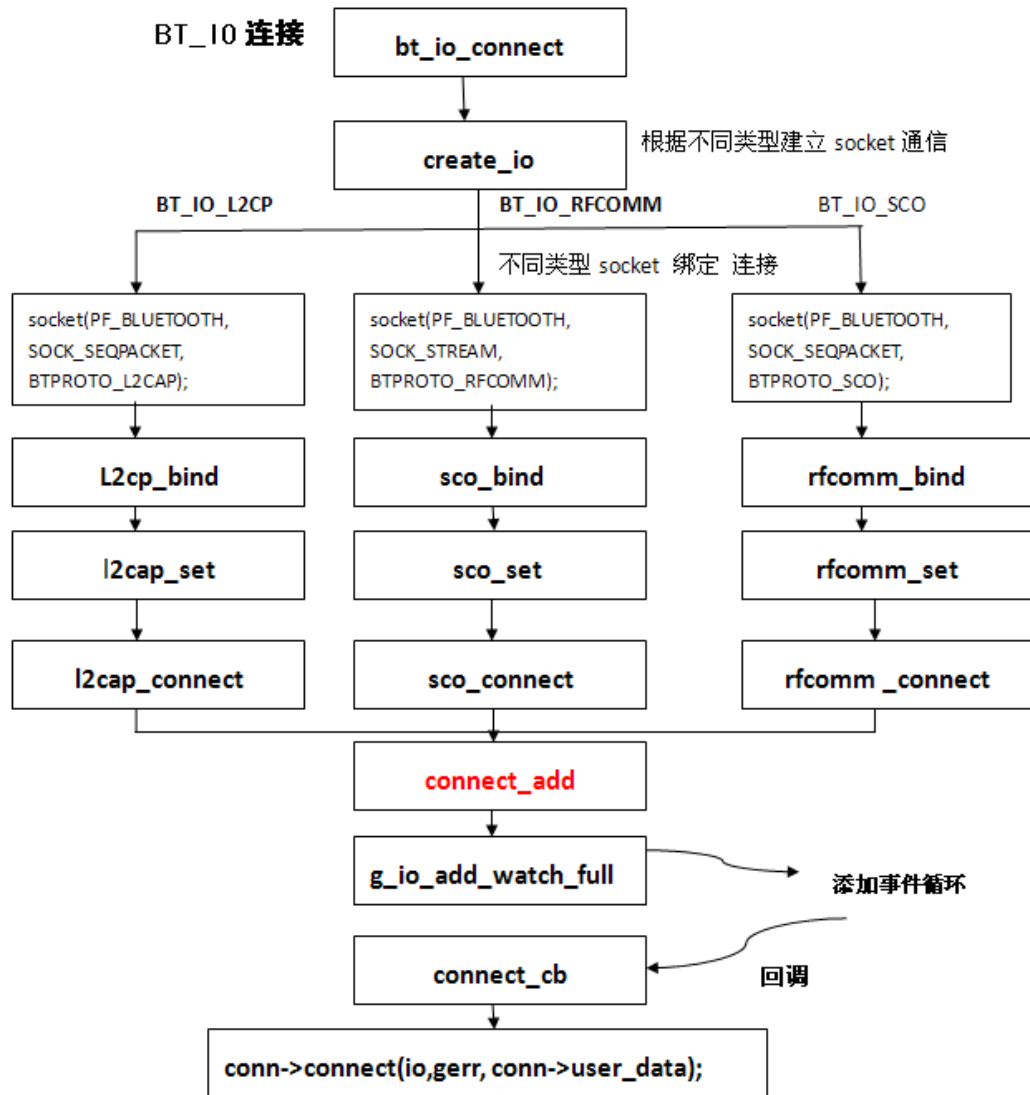
Figure 2.1: Bluetooth core system architecture

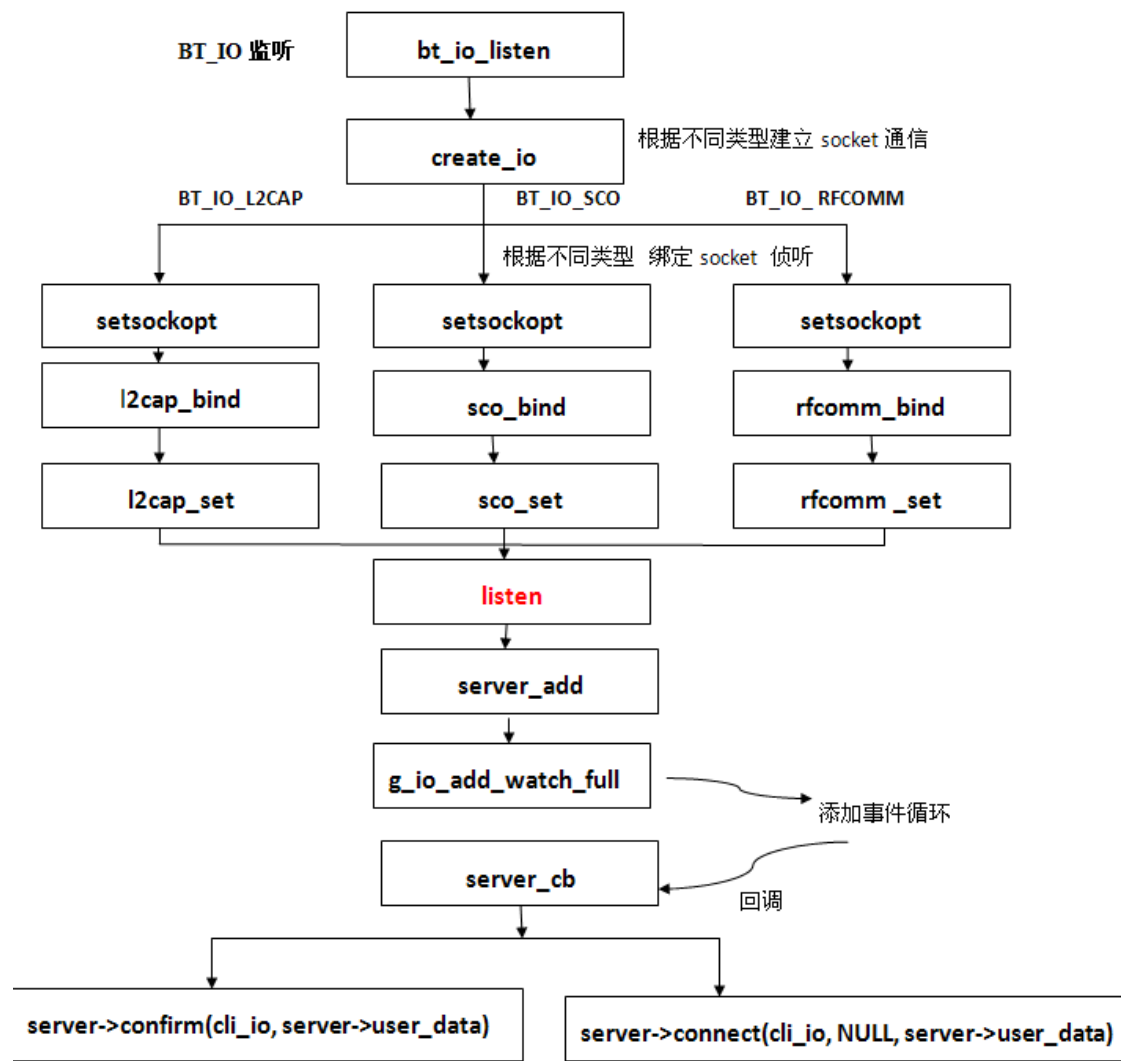
user space 模块包括:

- (1) bluetoothd
 - 1) 是 BlueZ 的中央守护进程
 - 2) 提供用于 UI 和其他子系统的 D-Bus 接口
- (2) 工具 (tools)
 - 1) bluetoothctl - bluetooth 命令行测试工具
 - 2) obexctl - obex 命令行测试工具
 - 3) hciconfig- HCI 信息跟踪 可作为 HCI 编程范例
 - 4) 其他用于测试, 开发和跟踪的命令行工具集
- (3) 上层协议
 - 1) Audio and media (**A2DP**, **AVRCP**)
 - 2) Telephony (**HFP**, HSP)
 - 3) Networking (PAN, 6LoWPAN)
 - 4) Input device (HID, HoG)
 - 5) OBEX (FTP, OPP, MAP, PBAP)
 - 6) Others

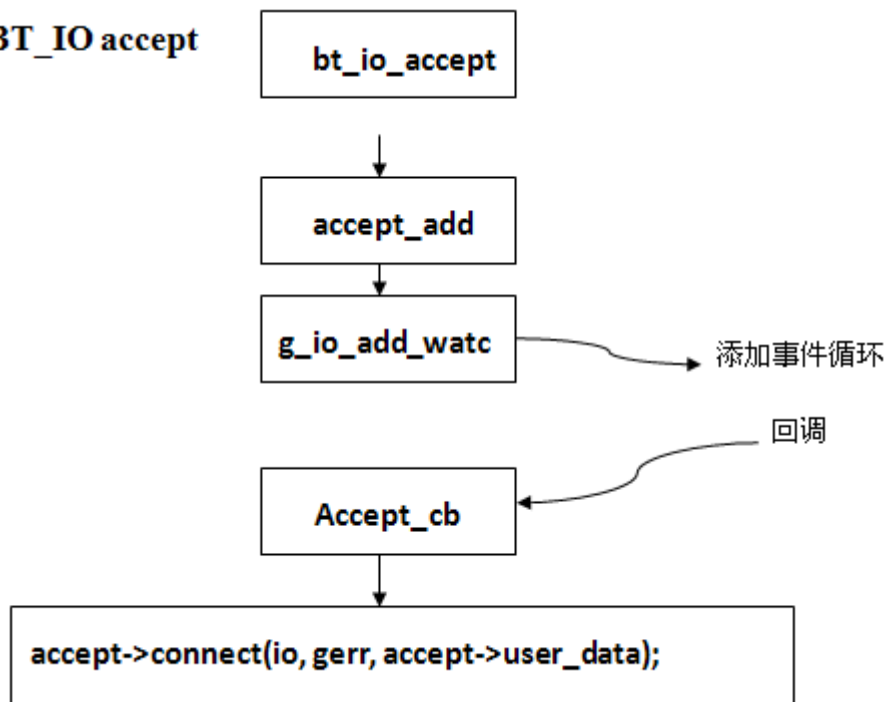


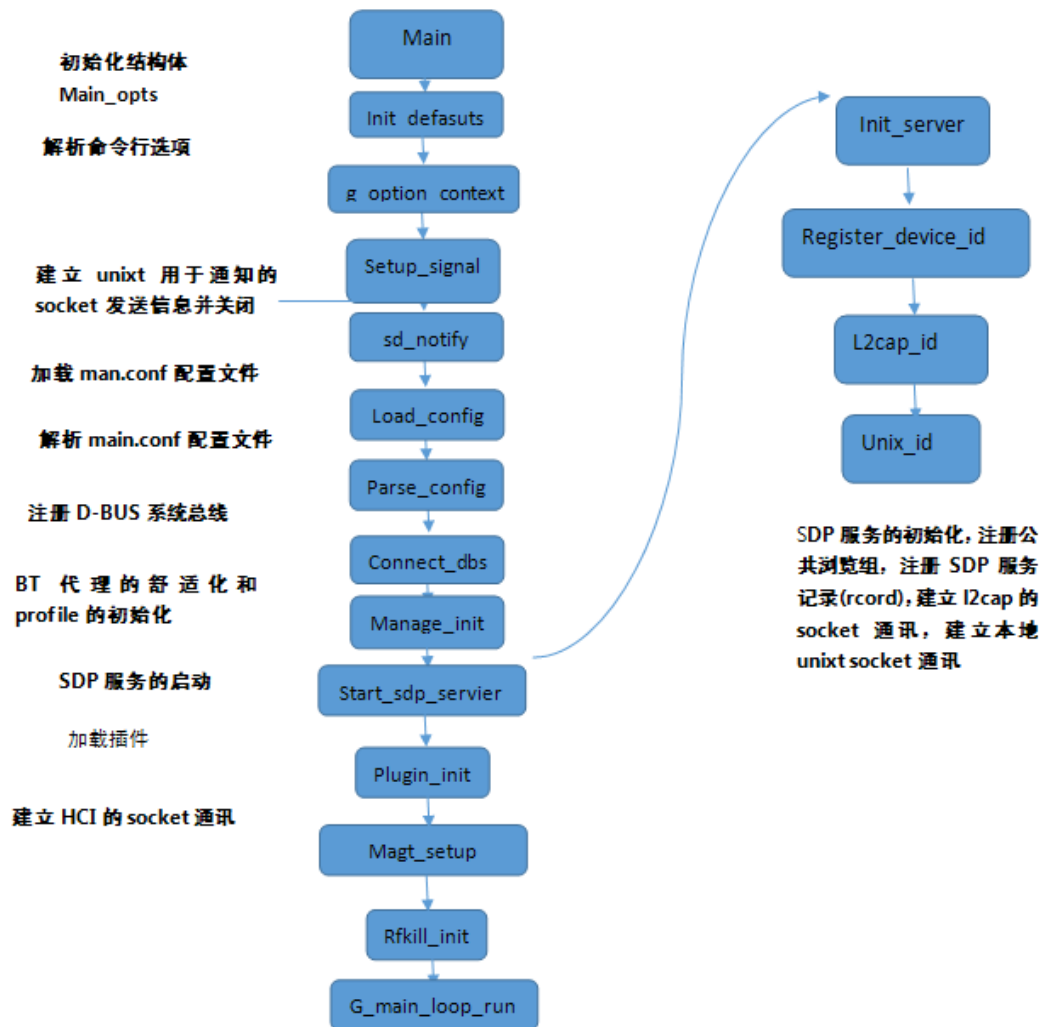
BT_IO 连接





BT_IO accept





1) A2DP AVRPC AVDTP 概念:

(1) AVDTP(AUDIO/VIDEO DISTRIBUTION TRANSPORT PROTOCOL)。是用来描述音频/视频在蓝牙设备间的传输的协议，是 A2DP 协议的基础协议，其在协议栈中的位置如下：AVDTP 协议建立在 connection-oriented L2CAP channel 上，只能支持 point-to-point signaling。

(2) Stream 代表两个 A/V 设备之间流多媒体数据的端到端的逻辑连接。

(3) Stream End Point (SEP): 应用程序通过这个接口提供 Transport Services and AV capabilities 来建立 Stream。

(4) Stream End Point Identifier (SEID): 标识 Stream End Point 的。

(5) Stream Context (SC): 两端设备都有的关于 Stream 的信息。

(6) Stream Handle (SH): 主要是暴露给用于程序使用的。

(7) Media Packets, Recovery Packets, and Reporting Packets: 根据三种不同的数据类型，有这三种数据包。其中，Basic Service 会用到 Basic service，Report service 会用到 Reporting Packets，Recovery service 会用到 Recovery Packets，此外 Multiplexing service 会用到 Media Packets 和其余一种或两种 Packet。

(8) Transport Session: 一条 stream 可以分解为多个 transport sessions，每个 transport session 对应一个 AVDTP 的 packet category，which means Media, Recovery, or Reporting packets。

(9) Transport Session Identifier (TSID): 标识 Transport Session。

(10) Transport Channel: 通常和一个 L2CAP Channel 对应。不用 AVDTP Multiplexing Mode 时，一条 Transport Channel 只传输一个 transport session；用 transport session 的情况下，一条 Transport Channel 可以传输多个 transport session (media,report 或者 recovery)。

(11) Transport Channel Identifier (TCID): 标识 Transport Channel，唯一关联一条 L2CAP channel。

2) A2DP 的连接:

AVDTP 连接的建立首先依赖于 L2CAP 连接的建立，它会在同一条 ACL Link 上建立两条 L2CAP Channel，一条是用来 Signaling，另一条用来进行 Stream，report 和 recovery 的传输。

2.1 Profile Stack

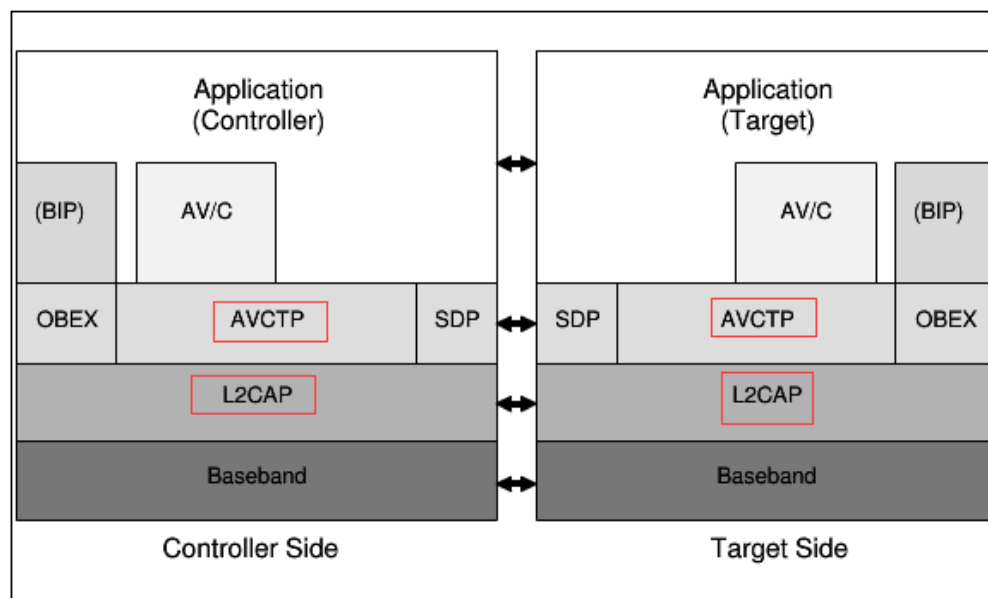


Figure 2.1: Protocol Model

Signaling 的主要过程为: 1.DISCOVER 2.GET_CAPABILITIES 3.SET_CONFIGURATION 4.OPEN 5.START。

| B... | Frame# | ACP... | Error C... | Addr. | INT SEID | Packet | Role | Signal ID | Tran... | Fra... | Delta |
|------|--------|--------|------------|-------|----------|---------------|--------|----------------------|---------|--------|--------------|
| | 220 | | | 1 | | Single Packet | Slave | DISCOVER | 10 | 11 | |
| | 244 | 1 | | 1 | | Single Packet | Master | DISCOVER | 10 | 13 | 00:00:00.0.. |
| | 256 | 1 | | 1 | | Single Packet | Slave | GET_ALL_CAPABILITIES | 11 | 12 | 00:00:00.0.. |
| | 257 | | | 1 | | Single Packet | Master | GET_ALL_CAPABILITIES | 11 | 23 | 00:00:00.0.. |
| | 265 | 1 | | 1 | 2 | Single Packet | Slave | SET_CONFIGURATION | 12 | 25 | 00:00:00.0.. |
| | 268 | | | 1 | | Single Packet | Master | SET_CONFIGURATION | 12 | 11 | 00:00:00.0.. |
| | 269 | 2 | | 1 | | Single Packet | Master | DELAYREPORT | 0 | 14 | 00:00:00.0.. |
| | 271 | 1 | | 1 | | Single Packet | Slave | OPEN | 13 | 12 | 00:00:00.0.. |
| | 273 | | | 1 | | Single Packet | Master | OPEN | 13 | 11 | 00:00:00.0.. |
| | 274 | | | 1 | | Single Packet | Slave | DELAYREPORT | 0 | 11 | 00:00:00.0.. |

2.1 SDP CLIENT-SERVER ARCHITECTURE

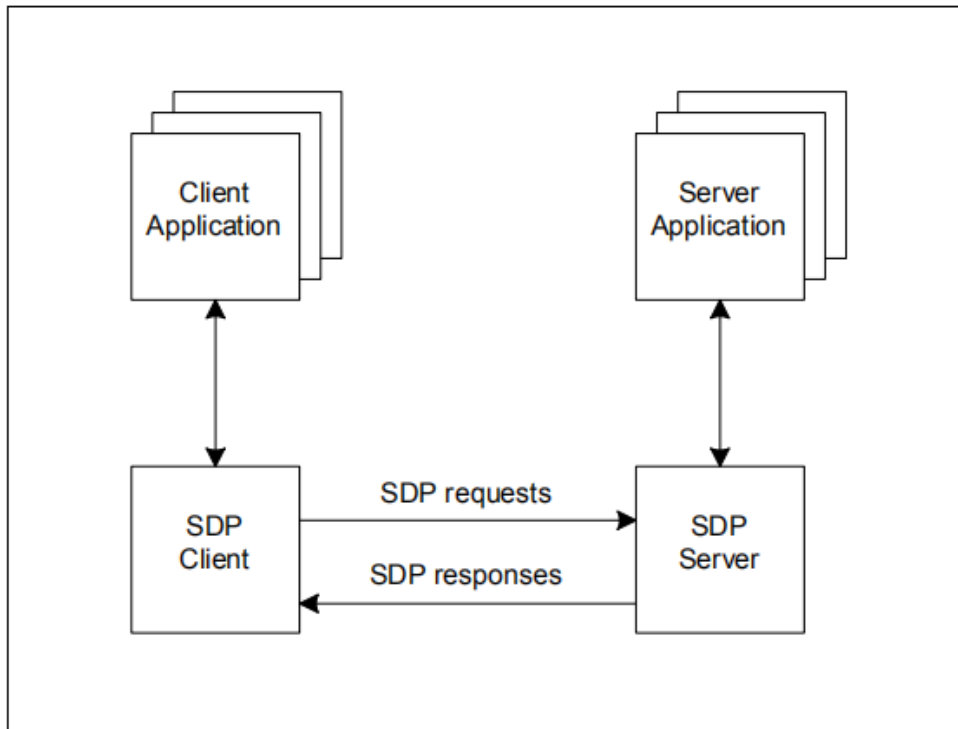


Figure 2.1: SDP Client-Server Interaction

SDP server 维护着一个服务条目(service record)列表.每个服务条目描述一个单独的服务属性。 SDP client 可以通过发送 SDP request 来得到服务条目。

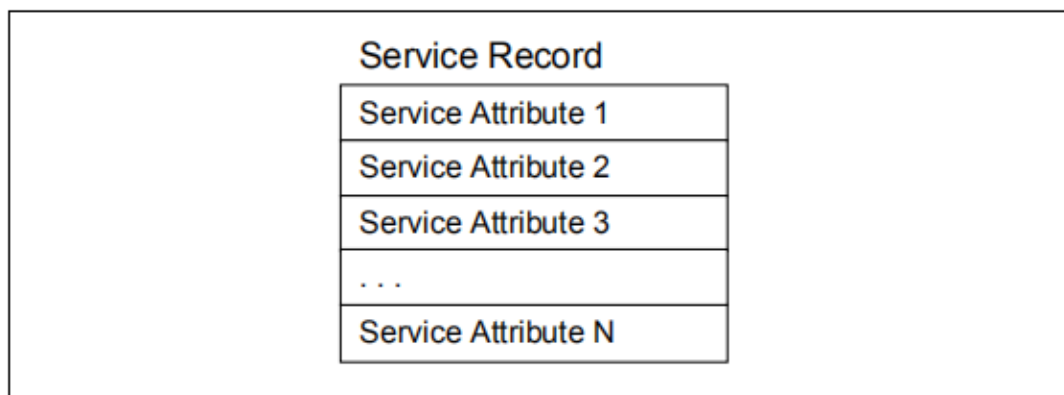


Figure 2.3: Service Record

所有的 Service 信息都包含于一个 Service Record 内。一个 Service Record 包含一个 Service attribute(Service 属性) list.

SDP 协议栈使用 request/response 模式工作，每个传输过程包括一个 request protocol data unit(PDU)和一个 response PDU. SDP 使用 L2CAP 连接传输数据

4.7 SERVICESEARCHATTRIBUTE TRANSACTION

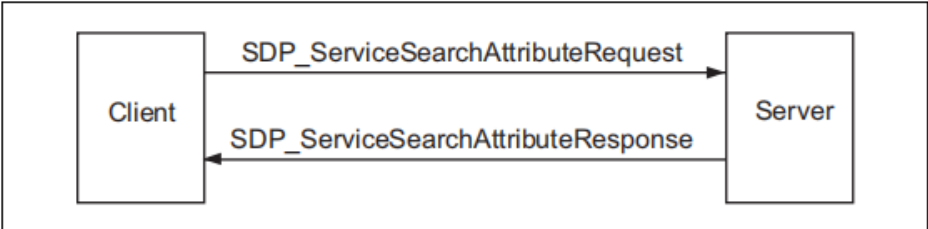
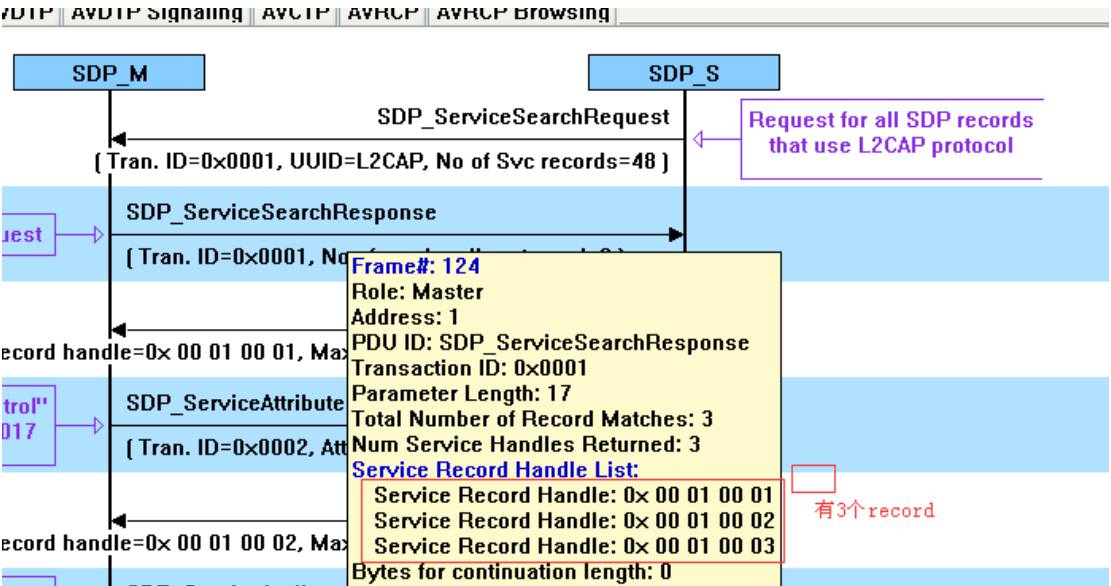


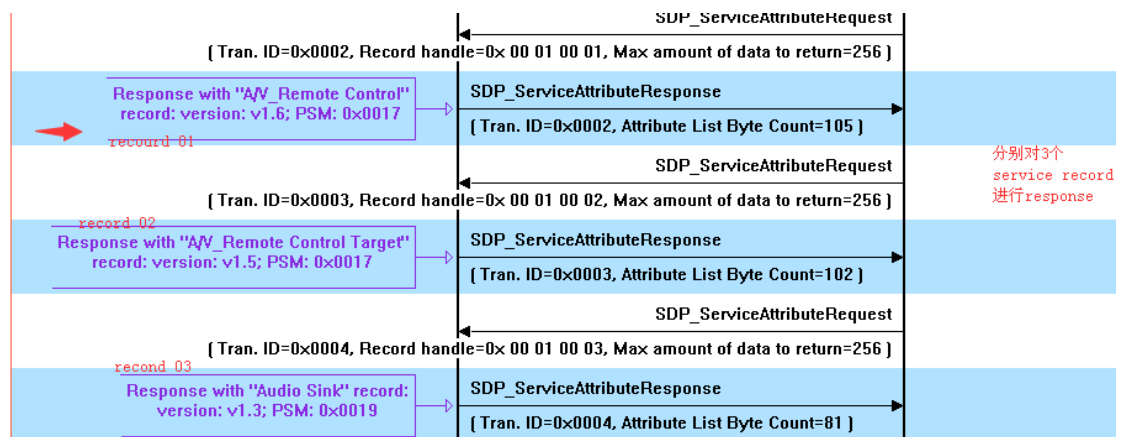
Figure 4.6: ServiceSearchAttribute Transaction

4.7.1 SDP_ServiceSearchAttributeRequest PDU

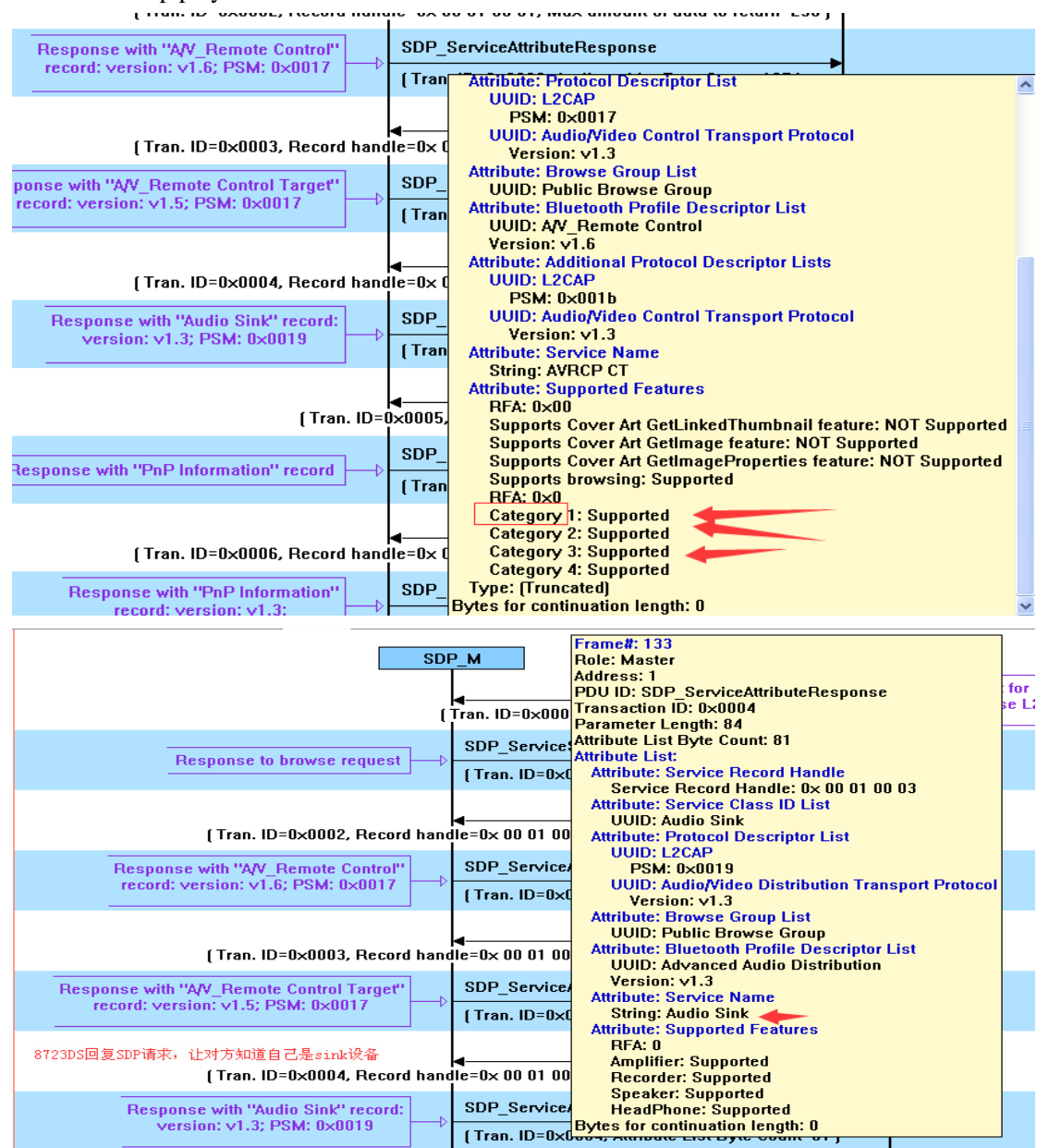
| PDU Type | PDU ID | Parameters |
|-----------------------------------|--------|---|
| SDP_ServiceSearchAttributeRequest | 0x06 | ServiceSearchPattern, MaximumAttributeByteCount, AttributeIDList, ContinuationState |

说明：PDU ID：用来识别 PDU。
TransactionID：用来识别 Request PUD 以及 Response PUD。并用来对比某个 Response PUD 是否对应着 Request PUD。
以 8723+配对 iphone 手机为例。手机发送 sdp req 请求流程如下：

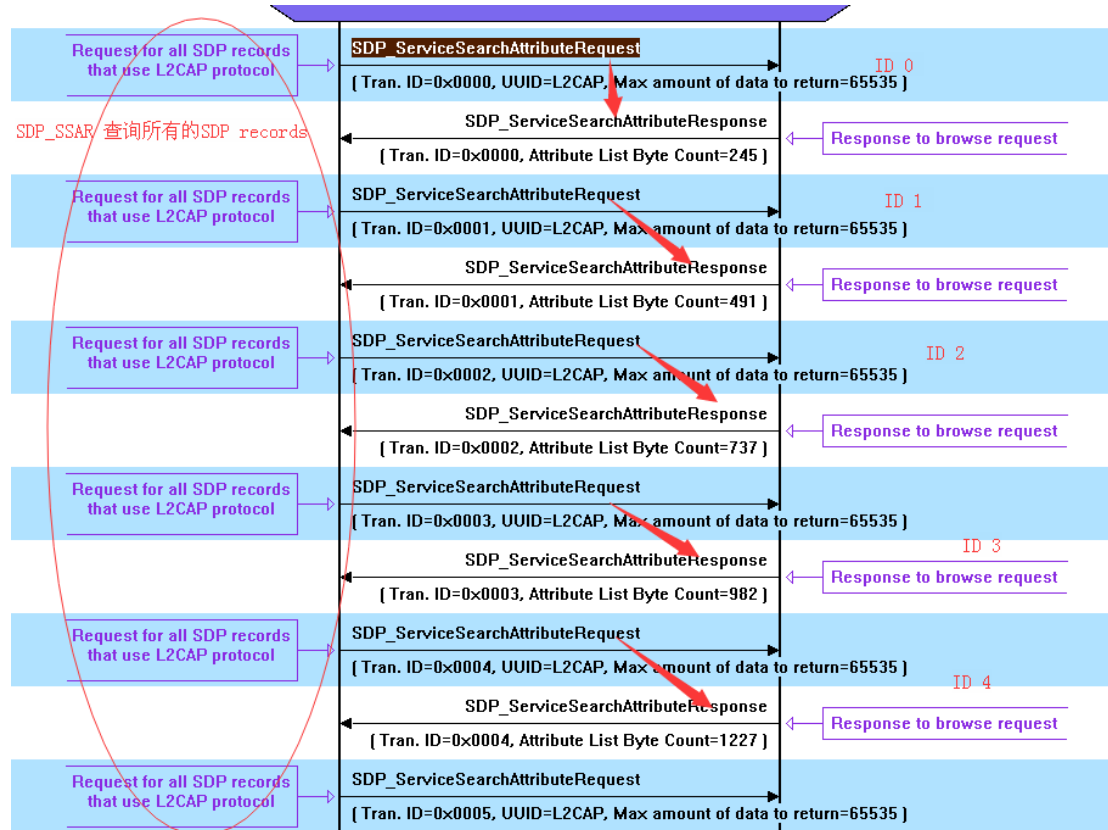




8723ds 端蓝牙回复 sdp req 告知手机，自己的设备类型和支持的 Category，协议栈里面用这个来建立 a2dp player



反过来，8723DS 也可以发送 SDP REQ 询问手机有哪些 sdp service



AVDTP :

AVDTP 协议建立在 connection-oriented L2CAP channel 上，只能支持 point-to-point signaling，详细可以看蓝牙 SPEC CORE

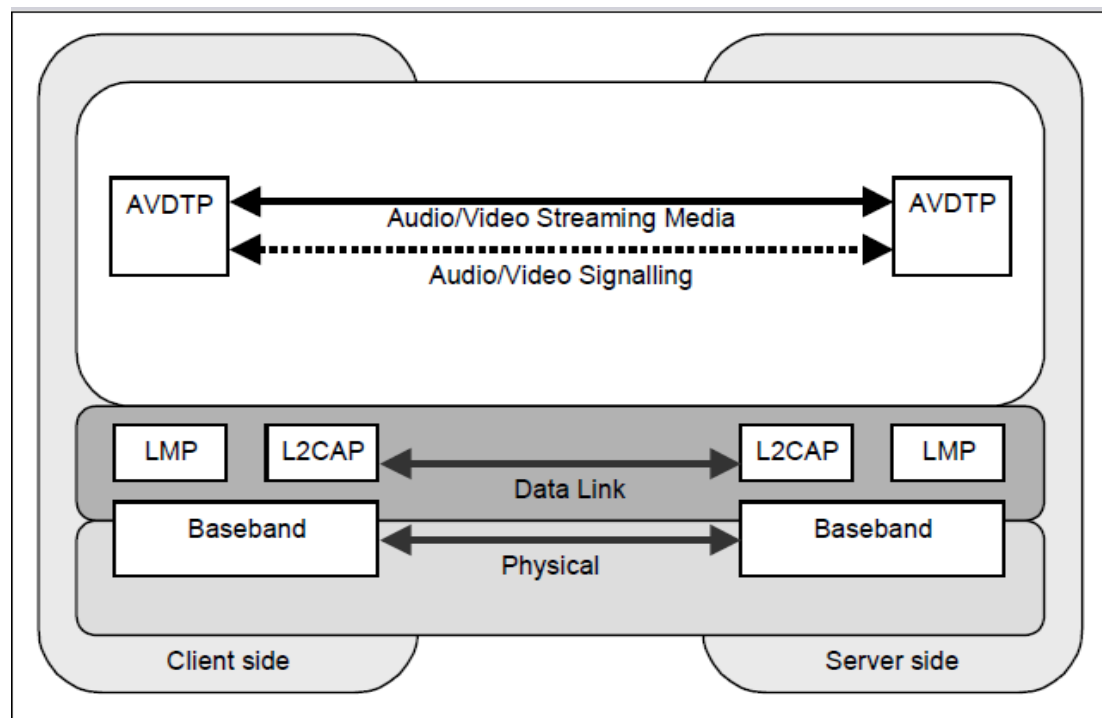


Figure 1.1: AVDTP and Bluetooth Protocol Stack

一些专业术语，对于看 SPEC 和蓝牙 SNIFF 报文有帮助：

Stream End Point Identifier (SEID): 标识 Stream End Point 的。

Transport Session: 一条 stream 可以分解为多个 transport sessions，每个 transport session 对应一个 AVDTP 的 packet category，which means Media, Recovery, or Reporting packets。

Transport Session Identifier (TSID): 标识 Transport Session。

Transport Channel: 通常和一个 L2CAP Channel 对应。不用 AVDTP Multiplexing Mode 时，一条 Transport Channel 只传输一个 transport session；用 transport session 的情况下，一条 Transport Channel 可以传输多个 transport session（media,report 或者 recovery）。

Transport Channel Identifier (TCID): 标识 Transport Channel，唯一关联一条 L2CAP channel。

stream End Point (**SEP**): 应用程序通过这个接口提供 Transport Services and AV capabilities 来建立 Stream，见 SPEC，一个完整的数据流控制 Signaling 过程：

| Unfiltered | | Errors | | | | | | | | | | | |
|------------|--------|--------|------------|-------|----------|---------------|-----------------|----------------------|----------------|--------|---------------|-------|--|
| HCI UART | | HCI | L2CAP | SDP | AVCTP | AVDTP | AVDTP Signaling | AVRCP | AVRCP Browsing | Data | | | |
| B... | Frame# | ACP... | Error C... | Addr. | INT SEID | Packet | Role | Signal ID | Tran... | Fra... | Delta | Times | |
| | 220 | | | 1 | | Single Packet | Slave | DISCOVER | 10 | 11 | | 2018- | |
| | 244 | 1 | | 1 | | Single Packet | Master | DISCOVER | 10 | 13 | 00:00:00.0... | 2018- | |
| | 256 | 1 | | 1 | | Single Packet | Slave | GET_ALL_CAPABILITIES | 11 | 12 | 00:00:00.0... | 2018- | |
| | 257 | | | 1 | | Single Packet | Master | GET_ALL_CAPABILITIES | 11 | 23 | 00:00:00.0... | 2018- | |
| | 265 | 1 | | 1 | 2 | Single Packet | Slave | SET_CONFIGURATION | 12 | 25 | 00:00:00.0... | 2018- | |
| | 268 | | | 1 | | Single Packet | Master | SET_CONFIGURATION | 12 | 11 | 00:00:00.0... | 2018- | |
| | 269 | 2 | | 1 | | Single Packet | Master | DELAYREPORT | 0 | 14 | 00:00:00.0... | 2018- | |
| | 271 | 1 | | 1 | | Single Packet | Slave | OPEN | 13 | 12 | 00:00:00.0... | 2018- | |
| | 273 | | | 1 | | Single Packet | Master | OPEN | 13 | 11 | 00:00:00.0... | 2018- | |
| | 274 | | | 1 | | Single Packet | Slave | DELAYREPORT | 0 | 11 | 00:00:00.0... | 2018- | |

Stream End Point Discovery: 远端设备提供支持的 SEP 列表和 media Type。

Get All Capabilities: Using the SEID as a reference, the local device can query the service capabilities of the remote SEP。

Stream Configuration: By referencing both the local and the remote SEID, the local device (the INT) configures the SEP of the remote device (the ACP)。

Stream Get Configuration: retrieve the last capabilities settings performed by the remote device on the SEP。

Stream Establishment: The opening of a transport session, while referencing the remote SEID, includes, if necessary, the establishment of a new Transport Channel。

Stream Start: After the Stream Establishment is complete, the Start Streaming procedure causes the streaming to start; i.e., Media (Reporting, Recovery) packets can be exchanged。

Stream Release: initiated by the upper layer within a device; a signal is sent indicating that a stream end-point be closed。

Stream Suspend: 挂起。

Stream Reconfigure: 对 Stream 进行配置。

Stream Reconfigure: 再次配置。

Security Control: 对 AV data transmitted over a Bluetooth 提供 protection。

Abort: 用于两个设备的同步。

General Reject: 对包含 invalid Signal Identifier 的 Command 进行的 response。

Delay Reporting: provide an initial delay report required for synchronized audio/video playback。

AVDTP:
 Role: Master
 Address: 1
 AVDTP Type: Signal

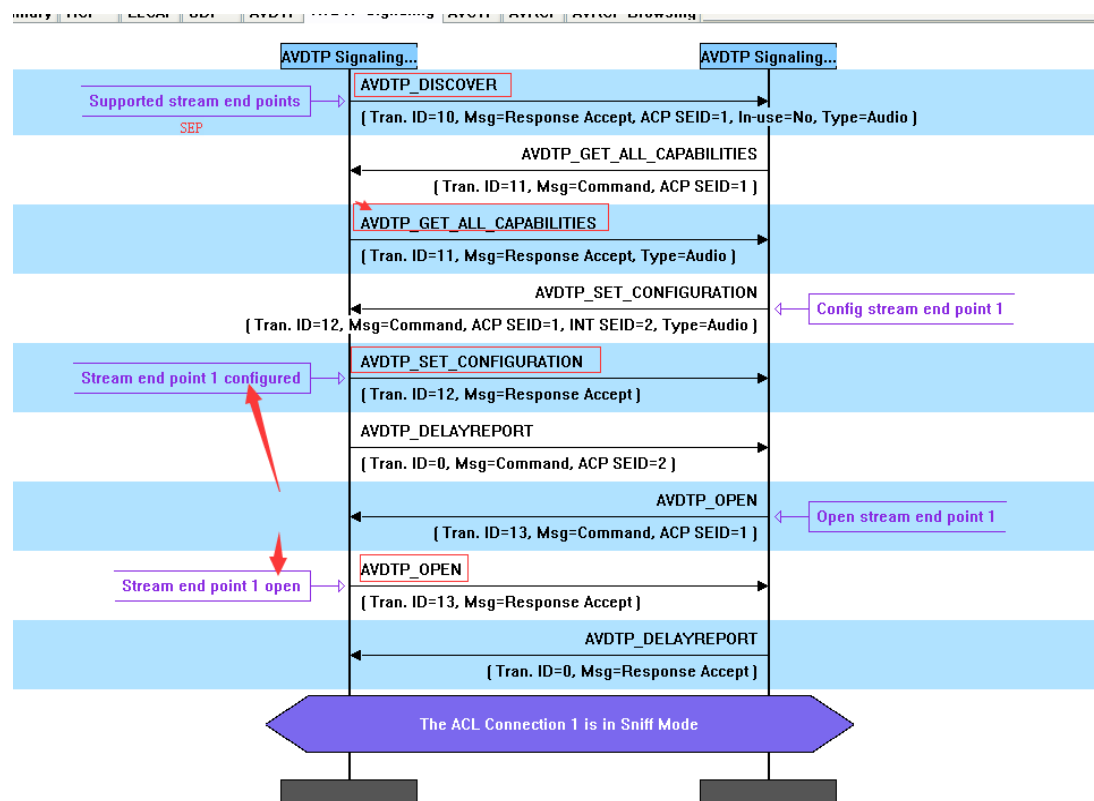
AVDTP Signaling: GET_ALL_CAP
 Role: Master
 Address: 1
 Transaction Label: 11
 Packet Type: Single Packet
 Message Type: Response Accept
 Signaling Identifier: AVDTP_GET_ALL_CAPABILITIES

Service Category: Media Transport
 Length Of Service Capability (LOSC): 0

Service Category: Media Codec
 Length Of Service Capability (LOSC): 6
 Media Type: Audio
 Media Codec Type: SBC

Codec Info Element
 Sampling Frequency(Hz) Supported
 16000
 32000
 44100
 48000
 Channel Mode Supported
 MONO
 DUAL CHANNEL
 STEREO
 JOINT STEREO
 Block Length Supported
 4
 8
 12
 16
 Subbands Supported
 4
 8
 Allocation Method Supported
 SNR
 Loudness
 Minimum Bitpool Value: 2
 Maximum Bitpool Value: 64

Service Category: Delay Reporting
 Length Of Service Capability (LOSC): 0



可以参考如下网址或者直接看 SPEC 文档，说的都是一样的东西：
<https://blog.csdn.net/xiaoxiaopengbo/article/details/51682346>
<https://blog.csdn.net/xubin341719/article/details/38335533>

AVRCP role 截图来至于蓝牙 core SPESDP 只提供侦测 Service 的机制，但不提供如何利用这些 Service 的机制，SDP 只提供侦测 Service 的办法，但如何用，SDP 不管，可以看到手机提供了比如短信，

C

2.2 Configuration and Roles

For the configuration examples for this profile, refer to the figures shown in Section 2.2.2.

2.2.1 Roles

The following roles are defined for devices that comply with this profile:

- The controller (CT) is a device that initiates a transaction by sending a command frame to a target. Examples for CT are a personal computer, a PDA, a mobile

phone, a remote controller or an AV device (such as an in car system, headphone, player/recorder, timer, tuner, monitor etc.).

- The target (TG) is a device that receives a command frame and accordingly generates a response frame. Examples for TG are an audio player/recorder, a video player/recorder, a TV, a tuner, an amplifier or a headphone.

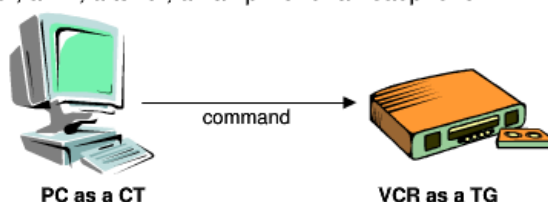


Figure 2.2: Controller and target

3.1 Feature Support

The table below shows the features requirements for this profile. Note that a device may have both CT and TG capabilities. In that case, features for both CT and TG are required.

| | Feature | Support in CT | Support in TG |
|-----|--|---------------|---------------|
| 1. | Connection establishment for control | M | O |
| 2. | Connection release for control | M | M |
| 3. | Connection establishment for browsing | C9 | C7 |
| 4. | Connection release for browsing | C9 | C10 |
| 5. | AV/C Info commands | O | M |
| 6. | Category 1: Player/Recorder | C3 | C3 |
| 7. | Category 2: Monitor/Amplifier | C3 | C3 |
| 8. | Category 3: Tuner | C3 | C3 |
| 9. | Category 4: Menu | C3 | C3 |
| 10. | Capabilities | O | C1 |
| 11. | Player Application Settings | O | O |
| 12. | Metadata Attributes for Current Media Item | O | C1 |
| 13. | Notifications | C2 | C2 |
| 14. | Continuation | C2 | C2 |

1. 播放/录音功能

支持播放或者录音设备的基本操作

2. 监视器/放大器功能

视频监视器或者音频放大器的基本操作

3. 调谐器功能

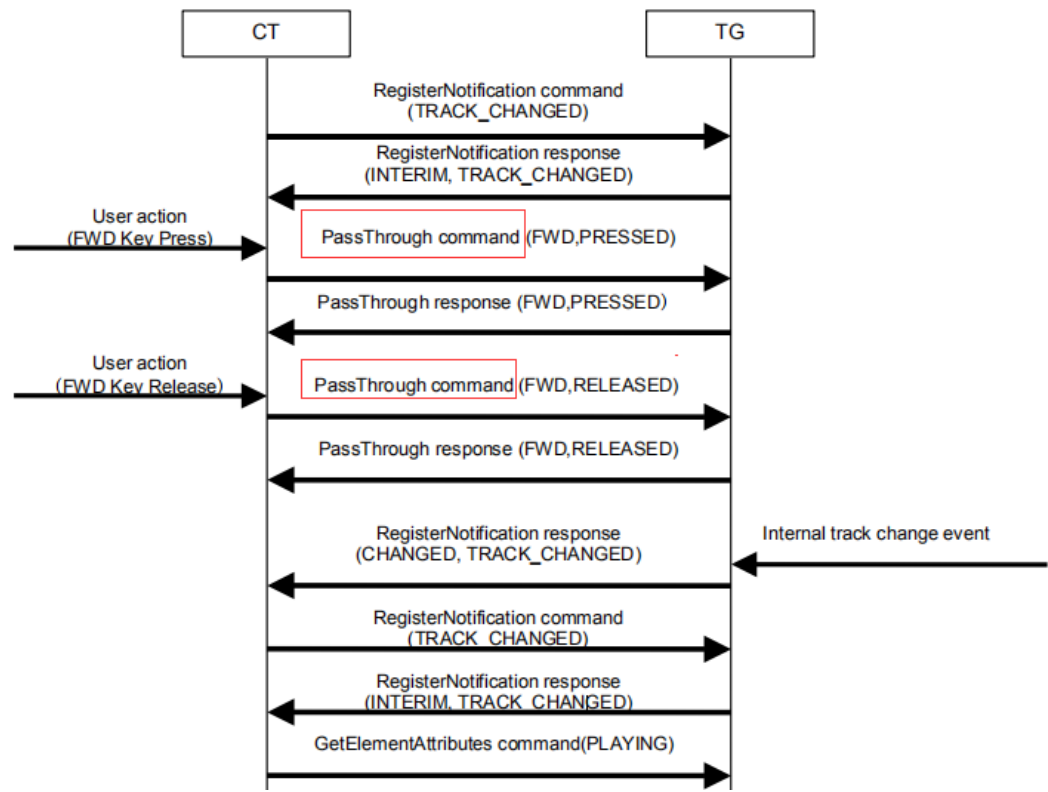
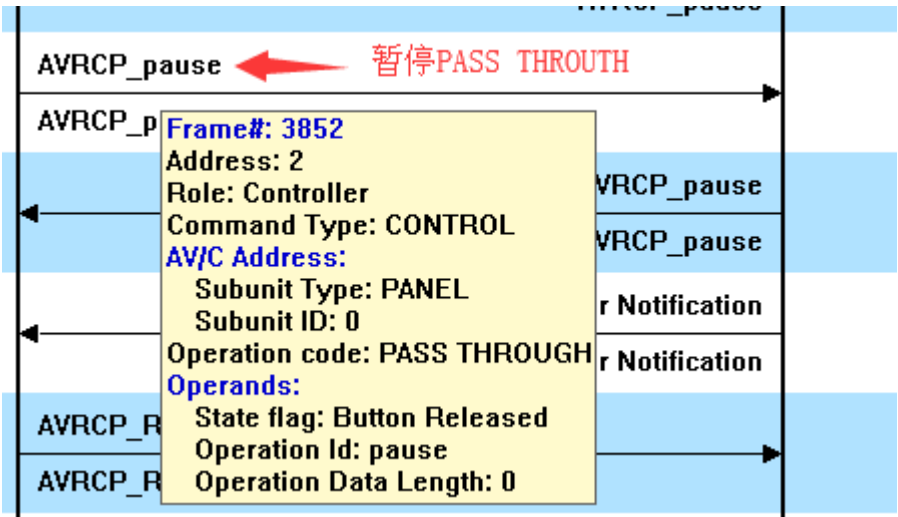
音视频调谐器的基本操作

4.菜单功能

这四类设备都需要支持歌曲控制功能。

歌曲控制：播放/暂停/停止/前一首/下一首。

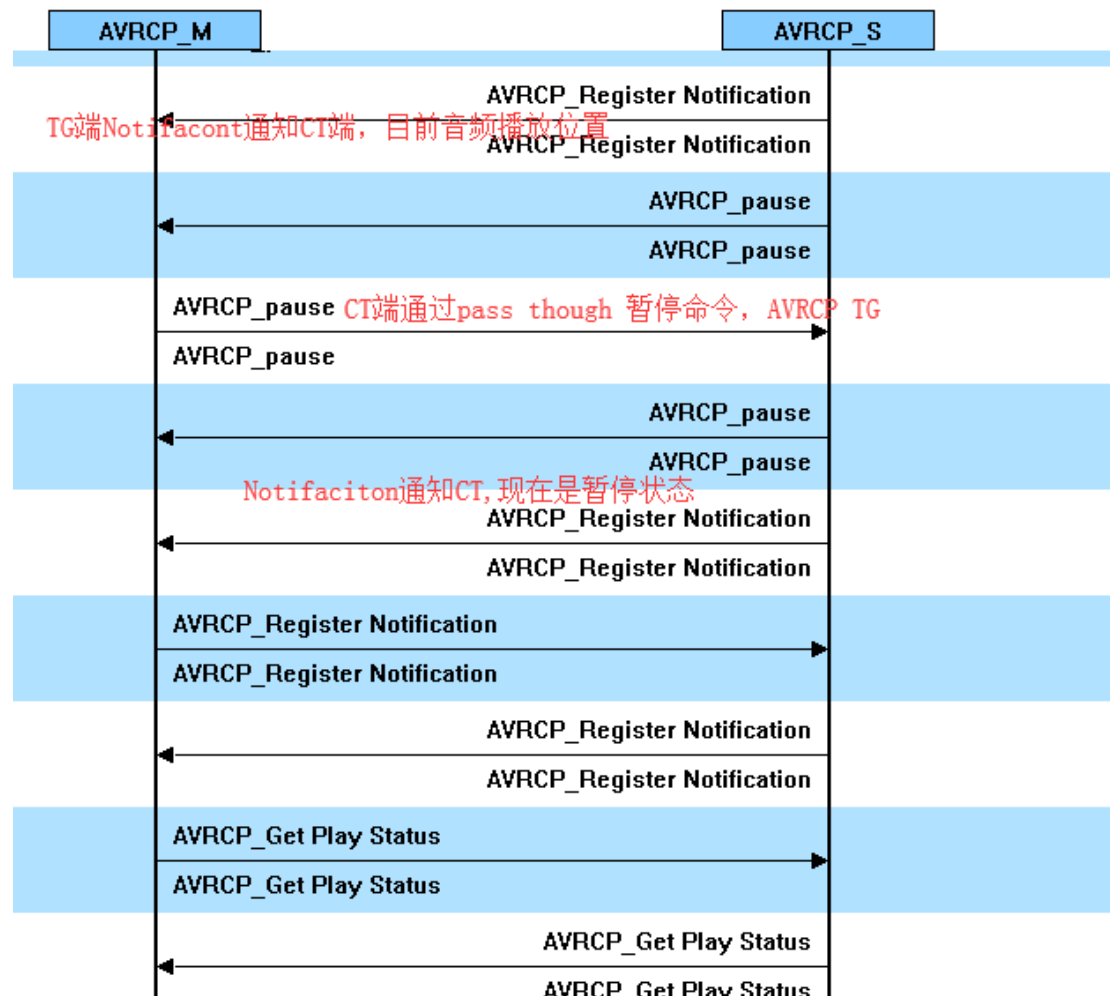
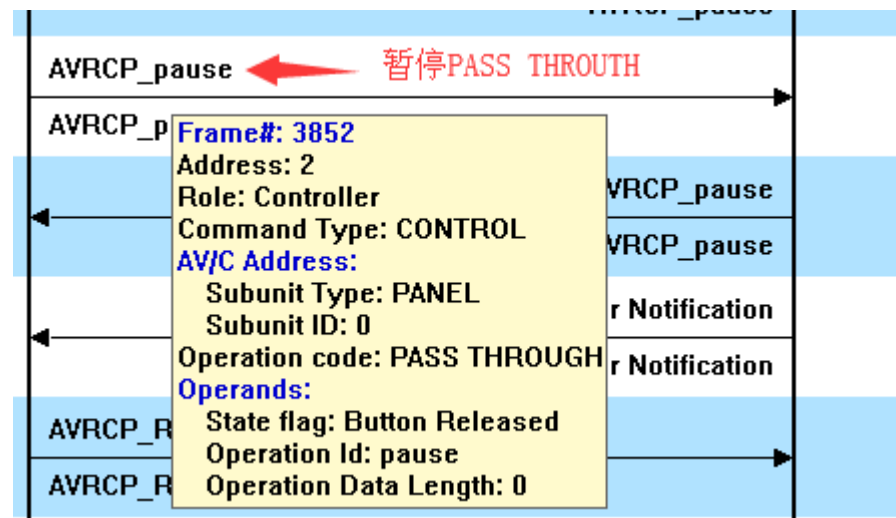
比如 PASSTHROUGH command



event notifications from the target

device 通知包括:

1. TG 端播放状态, 是 Playing Paused 等
2. Track change events



Response Formats (RegisterNotification)

Response Data format for EVENT_PLAYBACK_STATUS_CHANGED

| Parameters | Size (octets) | Description | Allowed Values |
|------------|---------------|--|---|
| EventID | 1 | Specific EventID | EVENT_PLAYBACK_STATUS_CHANGED (0x01) |
| PlayStatus | 1 | Indicates the current status of playback | 0x00: STOPPED 0x01: PLAYING 0x02: PAUSED 0x03: FWD_SEEK 0x04: REV_SEEK 0xFF: ERROR |

Table 6.31: Response EVENT_PLAYBACK_STATUS_CHANGED

| | | | | | | |
|-------|----|---|---------|------|------------------|-----------------------|
| 3.863 | CT | 1 | NOTIFY | | VENDOR-DEPENDENT | Register Notification |
| 3.867 | TG | 1 | INTERIM | | VENDOR-DEPENDENT | Register Notification |
| 3.868 | CT | 1 | STATUS | IMPL | | |
| 3.872 | TG | 1 | STATUS | IMPL | | |
| 3.879 | TG | 1 | CONTROL | ACCE | | |
| 3.829 | CT | 1 | CONTROL | ACCE | | |
| 3.834 | TG | 1 | CONTROL | ACCE | | |
| 3.835 | CT | 1 | CONTROL | ACCE | | |
| 3.839 | TG | 1 | CONTROL | ACCE | | |
| 3.862 | TG | 1 | CONTROL | ACCE | | |
| 3.863 | CT | 1 | NOTIFY | | | |
| 3.867 | TG | 1 | INTERIM | | | |
| 3.868 | CT | 1 | STATUS | IMPL | | |
| 3.873 | TG | 1 | STATUS | IMPL | | |
| 4.364 | CT | 1 | CONTROL | ACCE | | |
| 4.366 | TG | 1 | CONTROL | ACCE | | |
| 4.367 | CT | 1 | CONTROL | ACCE | | |
| 4.369 | TG | 1 | CONTROL | ACCE | | |
| 4.382 | TG | 1 | CONTROL | ACCE | | |
| 4.383 | CT | 1 | NOTIFY | | | |
| 4.388 | TG | 1 | STATUS | IMPL | | |
| 4.389 | CT | 1 | STATUS | IMPL | | |
| 4.393 | TG | 1 | CONTROL | ACCE | | |
| 4.758 | CT | 1 | CONTROL | ACCE | | |
| 4.762 | TG | 1 | CONTROL | ACCE | | |
| 4.763 | CT | 1 | CONTROL | ACCE | | |
| 4.768 | TG | 1 | CONTROL | ACCE | | |

Event Display - avrcp_test.cfa

| Event Number | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 |
|--------------|---|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|
| 2069665 | R | 02 | 01 | 20 | 13 | 00 | 0f | 00 | 41 | 00 | 32 | 11 | 0a | 0f | 48 | 00 |
| 2069681 | R | 02 | 01 | 20 | 13 | 00 | 0f | 00 | 41 | 00 | 32 | 11 | 0a | 0f | 48 | 00 |
| 2069697 | R | 02 | 01 | 20 | 13 | 00 | 0f | 00 | 41 | 00 | 32 | 11 | 0a | 0f | 48 | 00 |
| 2069713 | R | 02 | 01 | 20 | 13 | 00 | 0f | 00 | 41 | 00 | 32 | 11 | 0a | 0f | 48 | 00 |
| 2069729 | R | 02 | 01 | 20 | 13 | 00 | 0f | 00 | 41 | 00 | 32 | 11 | 0a | 0f | 48 | 00 |
| 2069745 | R | 02 | 01 | 20 | 13 | 00 | 0f | 00 | 41 | 00 | 32 | 11 | 0a | 0f | 48 | 00 |
| 2069761 | R | 02 | 01 | 20 | 13 | 00 | 0f | 00 | 41 | 00 | 32 | 11 | 0a | 0f | 48 | 00 |
| 2069777 | R | 02 | 01 | 20 | 13 | 00 | 0f | 00 | 41 | 00 | 32 | 11 | 0a | 0f | 48 | 00 |
| 2069793 | R | 02 | 01 | 20 | 13 | 00 | 0f | 00 | 41 | 00 | 32 | 11 | 0a | 0f | 48 | 00 |
| 2069809 | R | 02 | 01 | 20 | 13 | 00 | 0f | 00 | 41 | 00 | 32 | 11 | 0a | 0f | 48 | 00 |
| 2069825 | R | 02 | 01 | 20 | 13 | 00 | 0f | 00 | 41 | 00 | 32 | 11 | 0a | 0f | 48 | 00 |
| 2069841 | R | 02 | 01 | 20 | 13 | 00 | 0f | 00 | 41 | 00 | 32 | 11 | 0a | 0f | 48 | 00 |
| 2069857 | R | 02 | 01 | 20 | 13 | 00 | 0f | 00 | 41 | 00 | 32 | 11 | 0a | 0f | 48 | 00 |

异常表现 1: 使用网易播放器暂停播放命令（语音和按键表现不一样），发送了很多个才真正暂停，按键是正常的，使用语音暂停不一样

| | | | | | | | |
|--------|----|---|---------|----------|------------------|-----------------------|-----------------|
| 18.440 | CT | 1 | CONTROL | | PASS THROUGH | pause | Button Pushed |
| 18.446 | TG | 1 | CONTROL | ACCEPTED | PASS THROUGH | pause | Button Pushed |
| 18.447 | CT | 1 | CONTROL | | PASS THROUGH | pause | Button Released |
| 18.452 | TG | 1 | CONTROL | ACCEPTED | PASS THROUGH | pause | Button Released |
| 18.482 | CT | 1 | CONTROL | | PASS THROUGH | pause | Button Pushed |
| 18.487 | TG | 1 | CONTROL | ACCEPTED | PASS THROUGH | pause | Button Pushed |
| 18.488 | CT | 1 | CONTROL | | PASS THROUGH | pause | Button Released |
| 18.492 | TG | 1 | CONTROL | ACCEPTED | PASS THROUGH | pause | Button Released |
| 18.524 | CT | 1 | CONTROL | | PASS THROUGH | pause | Button Pushed |
| 18.530 | TG | 1 | CONTROL | ACCEPTED | PASS THROUGH | pause | Button Pushed |
| 18.531 | CT | 1 | CONTROL | | PASS THROUGH | pause | Button Released |
| 18.539 | TG | 1 | CONTROL | ACCEPTED | PASS THROUGH | pause | Button Released |
| 18.565 | CT | 1 | CONTROL | | PASS THROUGH | pause | Button Pushed |
| 18.573 | TG | 1 | CONTROL | ACCEPTED | PASS THROUGH | pause | Button Pushed |
| 18.574 | CT | 1 | CONTROL | | PASS THROUGH | pause | Button Released |
| 18.577 | TG | 1 | CONTROL | ACCEPTED | PASS THROUGH | pause | Button Released |
| 18.607 | CT | 1 | CONTROL | | PASS THROUGH | pause | Button Pushed |
| 18.614 | TG | 1 | CONTROL | ACCEPTED | PASS THROUGH | pause | Button Pushed |
| 18.615 | CT | 1 | CONTROL | | PASS THROUGH | pause | Button Released |
| 18.622 | TG | 1 | CONTROL | ACCEPTED | PASS THROUGH | pause | Button Released |
| 18.651 | CT | 1 | CONTROL | | PASS THROUGH | pause | Button Pushed |
| 18.655 | TG | 1 | CONTROL | ACCEPTED | PASS THROUGH | pause | Button Pushed |
| 18.656 | CT | 1 | CONTROL | | PASS THROUGH | pause | Button Released |
| 18.660 | TG | 1 | CONTROL | ACCEPTED | PASS THROUGH | pause | Button Released |
| 18.723 | TG | 1 | CHANGED | | VENDOR-DEPENDENT | Register Notification | Non-Fragmented |
| 18.729 | CT | 1 | NOTIFY | | VENDOR-DEPENDENT | Register Notification | Non-Fragmented |

异常的时候，看起来手机收到 PAUSE（ACCEPTED）后，没有任何动作，也没有改变为

pause 状态，是 ROM 控制逻辑错误？还是 dbus 协议栈处理第一个 PAUSE 命令不成功，重发了多个 PAUSE 命令？

正常交互是这样，但是对比 QQ 音乐和网易音乐，响应时间差太多，**网易响应时间要 1.07 秒，而 QQ 在 500 毫秒内，这个就是问题点了。结合猎户 SDK，找到问题原因。**

| | | | | | | | | | |
|--------|----|---|---------------|-----------------|------------------|-------|-----------------|-----------------------|----------------|
| 16,192 | CT | 1 | CONTROL | ← PAUSE press | PASS THROUGH | pause | Button Pushed | | |
| 16,197 | TG | 1 | ACCEPTED | | PASS THROUGH | pause | Button Pushed | | |
| 16,198 | CT | 1 | CONTROL | ← PAUSE release | PASS THROUGH | pause | Button Released | | |
| 16,203 | TG | 1 | ACCEPTED | | PASS THROUGH | pause | Button Released | | |
| 16,268 | TG | 1 | CHANGED | | VENDOR-DEPENDENT | | | Register Notification | Non-Fragmented |
| 16,269 | CT | 1 | NOTIFY | | VENDOR-DEPENDENT | | | Register Notification | Non-Fragmented |
| 16,273 | TG | 1 | INTERIM | | VENDOR-DEPENDENT | | | Register Notification | Non-Fragmented |
| 16,274 | CT | 1 | STATUS | | VENDOR-DEPENDENT | | | Get Play Status | Non-Fragmented |
| 16,277 | TG | 1 | IMPLEMENTE... | | VENDOR-DEPENDENT | | | Get Play Status | Non-Fragmented |

| | | | | | | |
|------------------|-------|-----------------------|--------------------|----|-----------------|--------------------------|
| PASS THROUGH | pause | Button Pushed | 15. 534931 下发PAUSE | 17 | 00:00:15.819358 | 2018/12/5 3:54:15.534931 |
| PASS THROUGH | pause | Button Pushed | | 17 | 00:00:00.055520 | 2018/12/5 3:54:15.590451 |
| PASS THROUGH | pause | Button Released | | 17 | 00:00:00.000386 | 2018/12/5 3:54:15.590837 |
| PASS THROUGH | pause | Button Released | 16. 608897 TG端状态改变 | 17 | 00:00:00.022386 | 2018/12/5 3:54:15.613223 |
| VENDOR-DEPENDENT | | Register Notification | | 24 | 00:00:00.995674 | 2018/12/5 3:54:16.608897 |
| VENDOR-DEPENDENT | | Register Notification | | 27 | 00:00:00.001688 | 2018/12/5 3:54:16.610565 |
| VENDOR-DEPENDENT | | Register Notification | | 24 | 00:00:00.019315 | 2018/12/5 3:54:16.629680 |
| VENDOR-DEPENDENT | | Get Play Status | | 22 | 00:00:00.002167 | 2018/12/5 3:54:16.632047 |
| VENDOR-DEPENDENT | | Get Play Status | | 31 | 00:00:00.068571 | 2018/12/5 3:54:16.700618 |

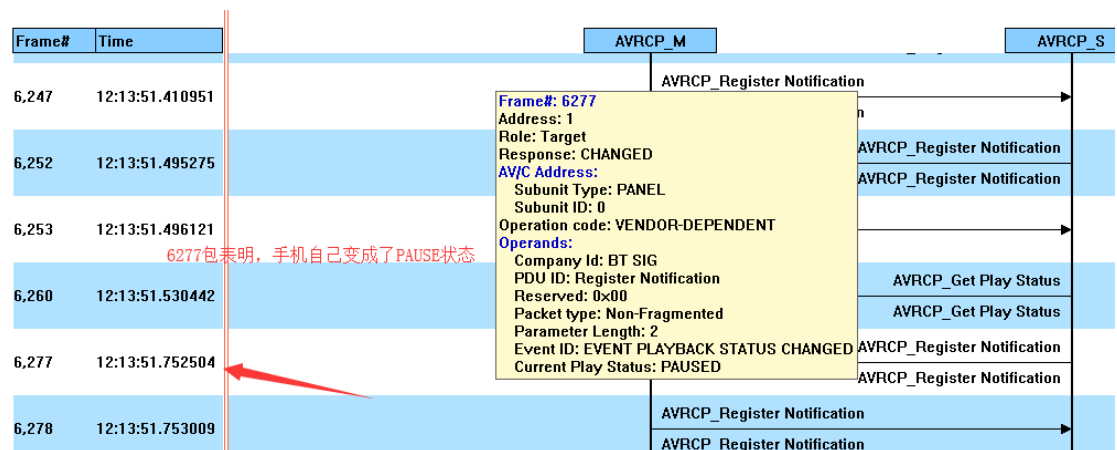
对比 QQ 音乐播放器：基本都是 500 毫秒内 APP 成功切换状态

| | | | | | | |
|------------------|-------|-----------------------|----------------------|----|-----------------|--------------------------|
| PASS THROUGH | pause | Button Pushed | 8. 912064 | 17 | 00:00:04.869083 | 2018/12/7 3:25:08.912064 |
| PASS THROUGH | pause | Button Pushed | | 17 | 00:00:00.080028 | 2018/12/7 3:25:08.992092 |
| PASS THROUGH | pause | Button Released | | 17 | 00:00:00.000369 | 2018/12/7 3:25:08.992461 |
| PASS THROUGH | pause | Button Released | 9. 383307 | 17 | 00:00:00.010205 | 2018/12/7 3:25:09.002666 |
| VENDOR-DEPENDENT | | Register Notification | | 24 | 00:00:00.380641 | 2018/12/7 3:25:09.383307 |
| VENDOR-DEPENDENT | | Register Notification | | 27 | 00:00:00.001432 | 2018/12/7 3:25:09.384739 |
| VENDOR-DEPENDENT | | Register Notification | | 24 | 00:00:00.044303 | 2018/12/7 3:25:09.429042 |
| VENDOR-DEPENDENT | | Get Play Status | | 22 | 00:00:00.000655 | 2018/12/7 3:25:09.429697 |
| VENDOR-DEPENDENT | | Get Play Status | | 31 | 00:00:00.094576 | 2018/12/7 3:25:09.524273 |
| PASS THROUGH | play | Button Pushed | QQ音乐播放器500毫秒内都可以返回成功 | 17 | 00:00:15.378203 | 2018/12/7 3:25:24.902476 |
| PASS THROUGH | play | Button Pushed | | 17 | 00:00:00.014028 | 2018/12/7 3:25:24.916504 |
| PASS THROUGH | play | Button Released | | 17 | 00:00:00.000615 | 2018/12/7 3:25:24.917119 |
| PASS THROUGH | play | Button Released | | 17 | 00:00:00.009414 | 2018/12/7 3:25:24.926533 |
| VENDOR-DEPENDENT | | Register Notification | | 24 | 00:00:00.125872 | 2018/12/7 3:25:25.052405 |
| VENDOR-DEPENDENT | | Register Notification | | 27 | 00:00:00.000436 | 2018/12/7 3:25:25.052841 |
| VENDOR-DEPENDENT | | Register Notification | | 24 | 00:00:00.010573 | 2018/12/7 3:25:25.063414 |
| VENDOR-DEPENDENT | | Get Play Status | | 22 | 00:00:00.000704 | 2018/12/7 3:25:25.064118 |
| VENDOR-DEPENDENT | | Get Play Status | | 31 | 00:00:00.028321 | 2018/12/7 3:25:25.092439 |
| PASS THROUGH | pause | Button Pushed | 32. 553393 | 17 | 00:00:07.460954 | 2018/12/7 3:25:32.553393 |
| PASS THROUGH | pause | Button Pushed | | 17 | 00:00:00.009439 | 2018/12/7 3:25:32.562832 |
| PASS THROUGH | pause | Button Released | | 17 | 00:00:00.000466 | 2018/12/7 3:25:32.563298 |
| PASS THROUGH | pause | Button Released | 32. 947017 | 17 | 00:00:00.008085 | 2018/12/7 3:25:32.571383 |
| VENDOR-DEPENDENT | | Register Notification | | 24 | 00:00:00.374634 | 2018/12/7 3:25:32.946017 |
| VENDOR-DEPENDENT | | Register Notification | | 27 | 00:00:00.000327 | 2018/12/7 3:25:32.946344 |

结论：手机收到 PAUSE 后，播放器没有及时做 PASUSE 处理，导致小雅 server 端还是认为是播放状态，进而继续发送 PAUSE，也就看到了如上发了很多个 PAUSE 命令，但是都没有得到播放器的响应，更新 STATE change，

解决办法：发送 PUASE 命令后，如果 PAUSE 失败，再延迟 500MS（暴力改法改成了 2S）再上报当前的状态，这样就可以保证播放器可以在 PASE 成功后才上报当前的播放器状态。
改问题系控制逻辑 BUG

异常表现 2: 使用网易播放器，从暂停时候，发送播放命令后，概率出现手机这边还是 PAUSE 状态。



```

bluetoothd[2403]: profiles/audio/avctp.c:avctp_passthrough_press() PLAY
bluetoothd[2403]: profiles/audio/avctp.c:avctp_passthrough_release() PLAY
bluetoothd[2403]: profiles/audio/avdtp.c:session_cb()
bluetoothd[2403]: profiles/audio/avdtp.c:avdtp_parse_cmd() Received START_CMD
bluetoothd[2403]: profiles/audio/a2dp.c:start_ind() Sink 0x36822630: Start_ind
bluetoothd[2403]: profiles/audio/avdtp.c:avdtp_ref() 0x36828b20: ref=3
bluetoothd[2403]: profiles/audio/avdtp.c:avdtp_sep_set_state() stream state changed: OPEN -> STREAMING
bluetoothd[2403]: profiles/audio/source.c:source_set_state() State changed /org/bluez/hci0/dev_D8_1D_72_1C_26_DC: SOURCE_STATE_CONNECTED -> SOURCE_STATE_PLAYING
bluetoothd[2403]: profiles/audio/transport.c:transport_update_playing() /org/bluez/hci0/dev_D8_1D_72_1C_26_DC/fd0 State=TRANSPORT_STATE_IDLE Playing=1
bluetoothd[2403]: profiles/audio/transport.c:transport_set_state() State changed /org/bluez/hci0/dev_D8_1D_72_1C_26_DC/fd0: TRANSPORT_STATE_IDLE -> TRANSPORT_STA
bluetoothd[2403]: profiles/audio/transport.c:media_owner_create() Owner created: sender=:1.2
bluetoothd[2403]: profiles/audio/a2dp.c:a2dp_sep_lock() SEP 0x36822630 locked
bluetoothd[2403]: profiles/audio/avdtp.c:avdtp_ref() 0x36828b20: ref=4
bluetoothd[2403]: profiles/audio/a2dp.c:setup_ref() 0x3683adb0: ref=1
bluetoothd[2403]: profiles/audio/avdtp.c:avdtp_unref() 0x36828b20: ref=3
bluetoothd[2403]: profiles/audio/transport.c:media_request_create() Request created: method=TryAcquire id=27
bluetoothd[2403]: profiles/audio/transport.c:media_owner_add() Owner :1.2 Request TryAcquire
bluetoothd[2403]: profiles/audio/transport.c:media_transport_set_owner() Transport /org/bluez/hci0/dev_D8_1D_72_1C_26_DC/fd0 Owner :1.2
bluetoothd[2403]: profiles/audio/transport.c:media_owner_remove() Owner :1.2 Request TryAcquire
bluetoothd[2403]: profiles/audio/transport.c:transport_set_state() State changed /org/bluez/hci0/dev_D8_1D_72_1C_26_DC/fd0: TRANSPORT_STATE_PENDING -> TRANSPORT_
bluetoothd[2403]: profiles/audio/a2dp.c:setup_unref() 0x3683adb0: ref=0
bluetoothd[2403]: profiles/audio/a2dp.c:setup_free() 0x3683adb0
bluetoothd[2403]: profiles/audio/avdtp.c:avdtp_unref() 0x36828b20: ref=2
bluetoothd[2403]: profiles/audio/player.c:media_player_set_status() playing
bluetoothd[2403]: profiles/audio/player.c:media_player_set_duration() 307000
bluetoothd[2403]: profiles/audio/player.c:media_player_set_position() 52146
bluetoothd[2403]: profiles/audio/player.c:media_player_set_status() playing
bluetoothd[2403]: profiles/audio/player.c:media_player_set_status() paused
bluetoothd[2403]: profiles/audio/player.c:media_player_set_duration() 307000
bluetoothd[2403]: profiles/audio/player.c:media_player_set_position() 52499
bluetoothd[2403]: profiles/audio/player.c:media_player_set_status() paused

```

发送pause命令, AVRCP手机端音频APP

手机接受命令, 并且切换为了playing, 播放了一段音频

但是手机又马上变成了pause状态

结论: 出问题时候的报文没有看到 PAUSE 包过来, 初步怀疑跟问题 1 相关, 就是说 PAUSE 命令延迟响应, 导致 PLAY 命令后, 中途又响应了之前的 CT 端发送的 PAUSE

关于蓝牙 ACL 连接的理解: 在 HCI 层 ACL Connection (蓝牙物理链路 ACL(Asynchronous Connectionless)) 的建立, 直接上干货

```

Frame 80: (Host) Len=11
  HCI UART:
    HCI Packet Type: Command Packet
  HCI:
    Packet from: Host
    HCI Command
      Opcode: 0x0409
      Group: Link Control
      Command: HCI_Accept_Connection_Request
      Total Length: 7
      Bluetooth Device Address: 0x38-6e-a2-d5-a7-0a
        LAP: 0xd5-a7-0a
        UAP: 0xa2
        NAP: 0x38-6e
      Role: Master

```

A2DP sink端先接受connect 请求

Bluetooth 扫描 扫描设备可以通过 hcitool hci0 scan 扫描设备

cmd_scan 命令下发后, 获取 hci 的信息, 通过 hci_inquiry 建立 sock 通讯, HCI 层 ioctl

```
nm_rsp = hci_inquiry(dev_id, length, num_rsp, lap, &info, flags)
```

cmd_scan -> socket(AF_BLUETOOTH, SOCK_RAW | SOCK_CLOEXEC, BTPROTO_HCI)

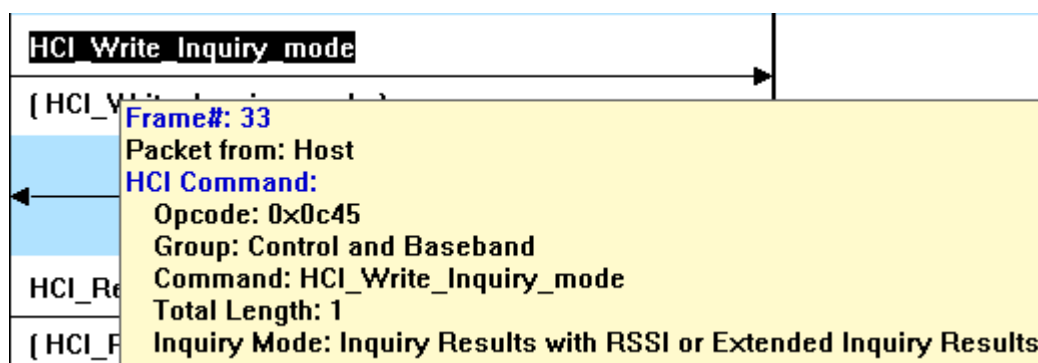
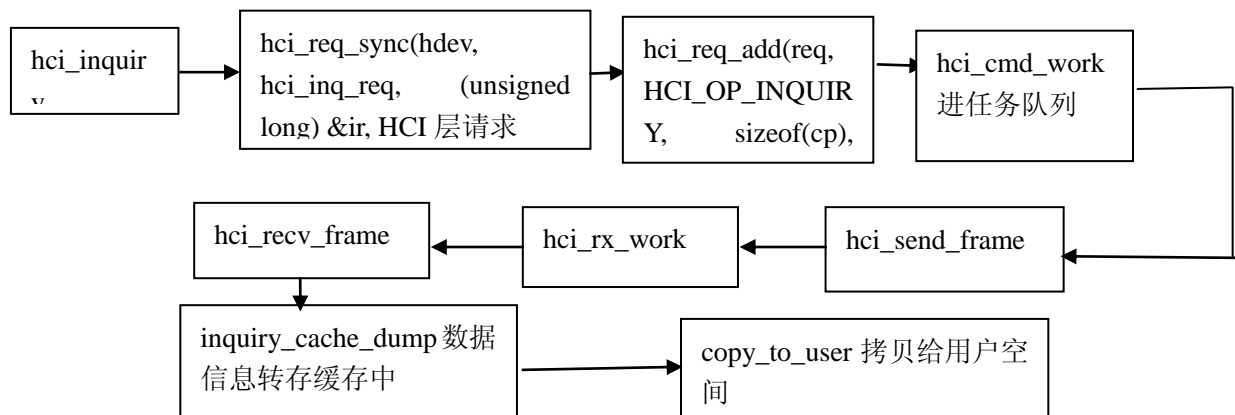
->ioctl(dd, HCIINQUIRY, (unsigned long) buf) 通过 socket 的 ioctl 调用到驱动层

也就是 Bluetooth driver hci_core kernel/net/Bluetooth

```
kernel/net/bluetooth$ vim hci_sock.c +735
```

Case HCIINQUIRY -> hci_inquiry(argp);

net/Bluetooth/ Bluetooth HCI Core



基于 BLE GATT 是基于 attribute 协议的 蓝牙配网

GATT 描述了如何使用 attribute 协议来发现、读、写和获取这些属性的标志，以及配置属性的广播。

client: 对于 service 启动命令和请求，可以接收 service 端的响应、指示和通知。

service: 接受来自 client 端的命令和请求，发送响应，指示和通知给 client 端。

例如：

client: 手机

service: 音箱

手机终端启动程序配置音箱 WIFI，或者读取音箱端 WIFI 连接是否成功的通知，音响提供 WIFI 特性的信息：例如 WIFI 服务和 WIFI service 下面其他被允许的可写的一些特性 characteristic 等。

用户需求

- 1、可改变的配置
- 2、发现设备上的服务和特性
- 3、读取一个特性值
- 4、写入一个特性值
- 5、通知的特性值
- 6、指示的特性值

通过 D-Bus API 控制蓝牙

BlueZ5 通过 D-Bus 提供 API 给第三方应用，所有的 BlueZ API 可以在 BlueZ5 源码的 doc/ 目录下查到，这些 API 可以通过 glib2.0 中的 GDBus 来访问，下面的网址提供了所有可用的 GDBus API:

<https://developer.gnome.org/gio/stable/index.html>

https://blog.csdn.net/iteye_17686/article/details/82036508

bluez 自带的 gdbus 库源码 位于 gdbus 目录

不管那么多，时间竟任务重，直接上代码，大体大体看到 bluez 代码基本都是这样的格式，后面具体在说一下。直接上干货


```

connection = g_dbus_setup_bus(DBUS_BUS_SYSTEM, NULL, NULL);
main_loop = g_main_loop_new(NULL, FALSE);
g_dbus_attach_object_manager(connection);
printf("gatt-service unique name: %s\n",
       dbus_bus_get_unique_name(connection));
#ifdef DUEOS
    dueros_socket_thread_create();
#endif

create_wifi_services();
client = g_dbus_client_new(connection, "org.bluez", "/");

main_loop = g_main_loop_new(NULL, FALSE);
dbus_conn = g_dbus_setup_bus(DBUS_BUS_SYSTEM, NULL, NULL);
g_dbus_attach_object_manager(dbus_conn);
client = g_dbus_client_new(dbus_conn, "org.bluez", "/org/bluez");

```

首先，必须要连接上 `DBusdbus_conn = g_dbus_setup_bus(DBUS_BUS_SYSTEM, NULL, NULL);`

其次，request a service on the bus，蓝牙 APP 都是如下方式：

```
GDBusClient client = g_dbus_client_new(dbus_conn, "org.bluez", "/org/bluez");
```

GDBusProxy 在蓝牙开发中，就是指注册到总线上的所有 interface，先注册比如：

```

BlueZ D-Bus Adapter API description
*****

Adapter hierarchy
=====
Service      org.bluez
Interface    org.bluez.Adapter1
Object path  [variable prefix]/{hci0,hci1,...}

```

一般来说，连接上 Dbus 和注册一个名称，应该是在程序最开始运行的时候就会进行的操作。当然，在程序的结束的时候，需要关闭掉与 Dbus 的连接。使用下面的函数：

```
g_dbus_client_unref(client);
```

```
dbus_connection_unref(dbus_conn);
```

org.bluez.Adapter1 -> adapter-api.txt

org.bluez.Device1 -> device-api.txt

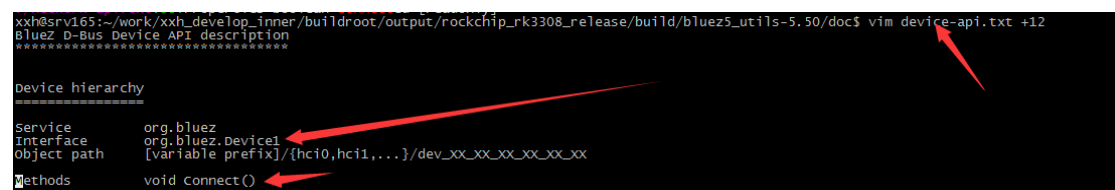
org.bluez.MediaPlayer1 -> media-api.txt

说明：如下代码都是实际项目 APP 的使用代码，借鉴的是 bluez 里面的 dbus 代码。

通常用 g_dbus_proxy_method_call 调用一个远程方法（remote method），

```
gboolean g_dbus_proxy_method_call(GDBusProxy *proxy, const char *method,
                                   GDBusSetupFunction setup,
                                   GDBusReturnFunction function, void *user_data,
                                   GDBusDestroyFunction destroy)
```

ARRCP 和 BT connect 都用到了这个方法调用 org.bluez.Device1(device-api.txt) 的 connect 方法，



以及 org.bluez.MediaPlayer1 的 play/pause 等方法（media-api.txt）

说明一个总要地方：

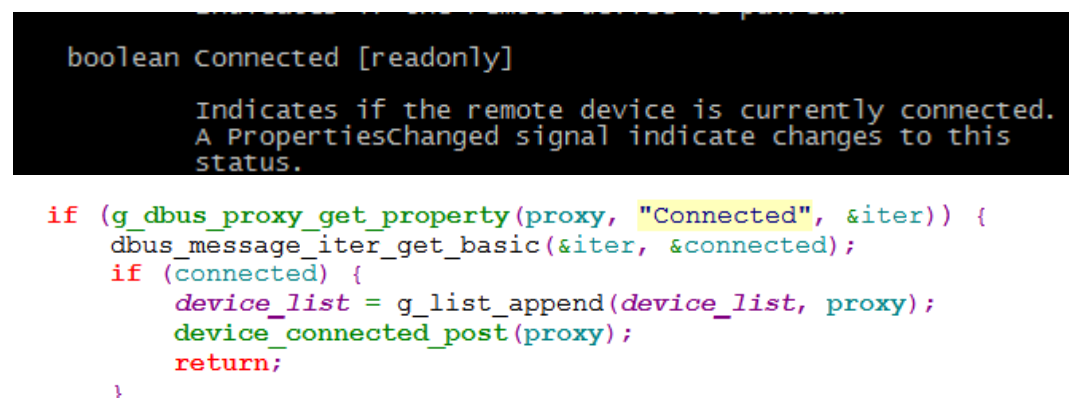
```
g_dbus_client_set_proxy_handlers(client, proxy_added, proxy_removed,
                                  property_changed, NULL);
```

property_changed 这个函数，一般在 APP 开发用来获取蓝牙的 disconnect 、 connect（），power，

可以在自己的 property_changed 中灵活处理响应 interface 的状态

g_dbus_proxy_get_property 能够获取到当期 interface 的各类状态，（只要 API 里面有定义的）

比如：device 里面定义的：



```
boolean Connected [readonly]
    Indicates if the remote device is currently connected.
    A Propertieschanged signal indicate changes to this
    status.

if (g_dbus_proxy_get_property(proxy, "Connected", &iter)) {
    dbus_message_iter_get_basic(&iter, &connected);
    if (connected) {
        device_list = g_list_append(device_list, proxy);
        device_connected_post(proxy);
        return;
    }
}
```

比如：Adapter1 里面的

```
boolean Powered [readwrite]
```


Switch an adapter on or off. This will also set the appropriate connectable state of the controller.

The value of this property is not persistent. After restart or unplugging of the adapter it will reset back to false.

```
static int adapter_is_powered(GDBusProxy *proxy)
{
    DBusMessageIter iter;
    dbus_bool_t powered;

    if (g_dbus_proxy_get_property(proxy, "Powered", &iter)) {
        dbus_message_iter_get_basic(&iter, &powered);
        if (powered)
            return 1;
    }

    return 0;
}
```



比如: MediaPlayer1 :

```
string Status [readonly]
```

Possible status: "playing", "stopped", "paused",
"forward-seek", "reverse-seek"
or "error"

```
proxy = default_player;
if (g_dbus_proxy_get_property(proxy, "Status", &iter) == FALSE)
    return AVRCP_PLAY_STATUS_ERROR; //unkonw status
dbus_message_iter_get_basic(&iter, &valstr);
if (!strcasecmp(valstr, "stopped"))
    return AVRCP_PLAY_STATUS_STOPPED;
else if (!strcasecmp(valstr, "playing"))
    return AVRCP_PLAY_STATUS_PLAYING;
else if (!strcasecmp(valstr, "paused"))
    return AVRCP_PLAY_STATUS_PAUSED;
else if (!strcasecmp(valstr, "forward-seek"))
    return AVRCP_PLAY_STATUS_FWD_SEEK;
```

```
GDBusProxy *proxy = (GDBusProxy *)user_data;
DBusMessageIter iter;

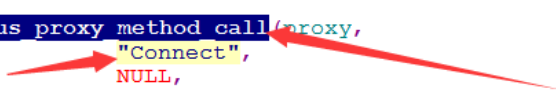
if (reconnect_timer) {
    g_source_remove(reconnect_timer);
    reconnect_timer = 0;
}

if (!proxy) {
    error("Invalid proxy, stop reconnecting");
    return FALSE;
}

if (g_list_length(device_list) > 0) {
    error("Device already connected");
    return FALSE;
}

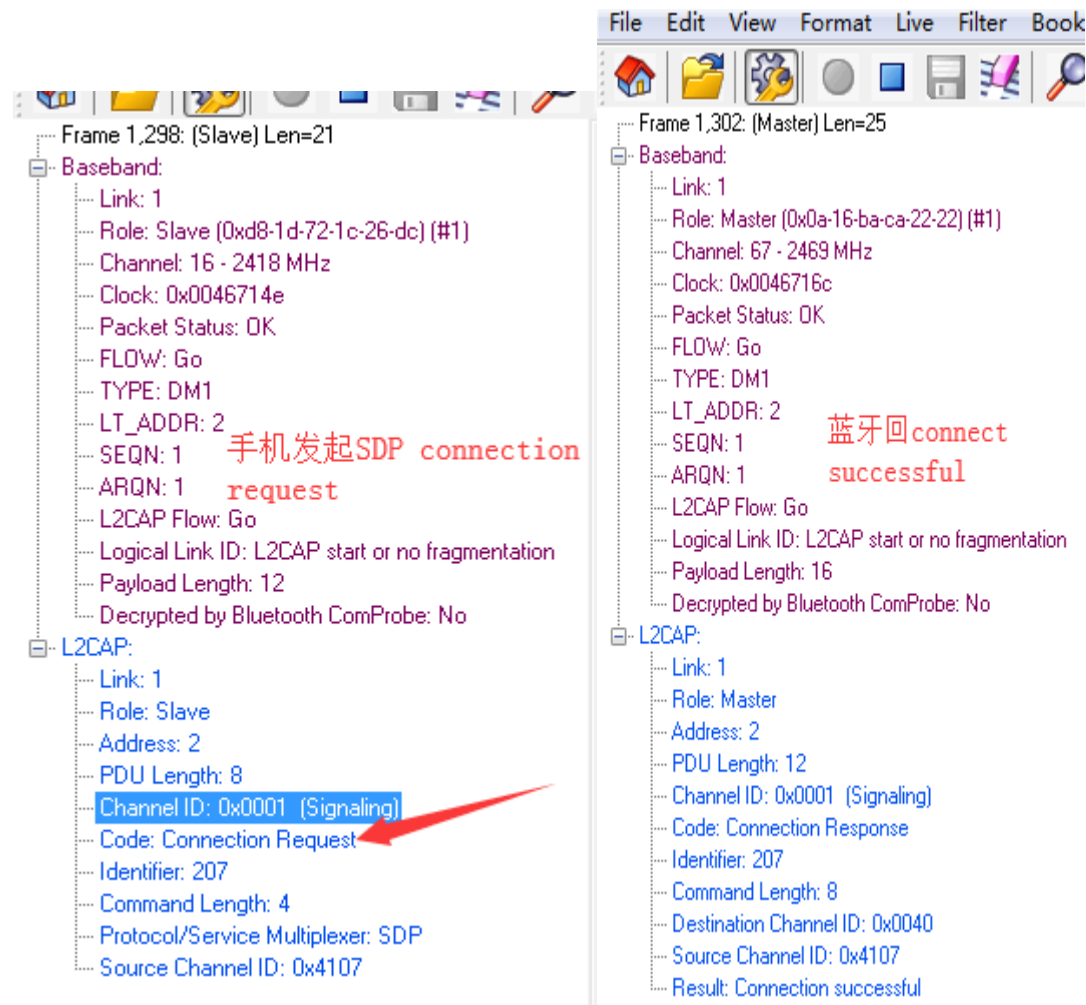
pr_info("Connect target device: %s", g_dbus_proxy_get_path(proxy));

if (g_dbus_proxy_method_call(proxy,
    "Connect",
    NULL,
    reconn_device_reply,
    user_data, NULL) == FALSE) {
    error("Failed to call org.bluez.Device1.Connect");
}
```



蓝牙 BUG 的记录:

问题 1: 小米 6 手机 ARVRP player 连接异常情况
先来看正常的初始连接过程:



The image displays two Wireshark packet capture windows side-by-side, showing the initial Bluetooth connection process. The left window shows Frame 1,298 (Slave) with a length of 21 bytes. The right window shows Frame 1,302 (Master) with a length of 25 bytes. Both frames are categorized under 'Baseband' and 'L2CAP'.

Frame 1,298: (Slave) Len=21

- Baseband:
 - Link: 1
 - Role: Slave (0xd8-1d-72-1c-26-dc) (#1)
 - Channel: 16 - 2418 MHz
 - Clock: 0x0046714e
 - Packet Status: OK
 - FLOW: Go
 - TYPE: DM1
 - LT_ADDR: 2
 - SEQN: 1
 - ARQN: 1
 - L2CAP Flow: Go
 - Logical Link ID: L2CAP start or no fragmentation
 - Payload Length: 12
 - Decrypted by Bluetooth ComProbe: No
- L2CAP:
 - Link: 1
 - Role: Slave
 - Address: 2
 - PDU Length: 8
 - Channel ID: 0x0001 (Signaling)
 - Code: Connection Request
 - Identifier: 207
 - Command Length: 4
 - Protocol/Service Multiplexer: SDP
 - Source Channel ID: 0x4107

Frame 1,302: (Master) Len=25

- Baseband:
 - Link: 1
 - Role: Master (0x0a-16-ba-ca-22-22) (#1)
 - Channel: 67 - 2469 MHz
 - Clock: 0x0046716c
 - Packet Status: OK
 - FLOW: Go
 - TYPE: DM1
 - LT_ADDR: 2
 - SEQN: 1
 - ARQN: 1
 - L2CAP Flow: Go
 - Logical Link ID: L2CAP start or no fragmentation
 - Payload Length: 16
 - Decrypted by Bluetooth ComProbe: No
- L2CAP:
 - Link: 1
 - Role: Master
 - Address: 2
 - PDU Length: 12
 - Channel ID: 0x0001 (Signaling)
 - Code: Connection Response
 - Identifier: 207
 - Command Length: 8
 - Destination Channel ID: 0x0040
 - Source Channel ID: 0x4107
 - Result: Connection successful

Red annotations are present: '手机发起SDP connection request' (Mobile initiates SDP connection request) next to the Slave's SEQN and ARQN, and '蓝牙回connect successful' (Bluetooth returns connect successful) next to the Master's SEQN and ARQN. A red arrow points to the 'Code: Connection Request' field in the Slave's L2CAP section.

异常情况描述, 但是 AVRCP 交互有异常, 但是 iphone 交互正常, android 小米 6 手机 SDP 交互信息不正常, 分析原因: 8723 做 sdp request 得到的 service 相关 feacher, 的确看到 8723 没有主动做 sdp req, 只有手机发 sdp req 蓝牙芯片 (8723ds) 回复, 按照如下方式添加打印, 发现 versuon 和 features 不正常的时候有问题, 因为这个 feature 是从 (8723) 获取到的 sdp record 的信息中提取出来

profiles/audio/avrcp.c

```

static void controller_init(struct avrcp *session)
{
    struct avrcp_player *player;
    struct btd_service *service;
    struct avrcp_data *controller;

    if (session->controller != NULL)
        return;

    controller = data_init(session, AVRCP_TARGET_UUID);
    session->controller = controller;

    DBG("%p version 0x%04x", controller, controller->version);

    service = btd_device_get_service(session->dev, AVRCP_TARGET_UUID);
    btd_service_connecting_complete(service, 0);

    /* only create player if category 1 is supported */
    if (controller->features & AVRCP_FEATURE_CATEGORY_1) {
        DBG("====Debug player====");
        player = create_ct_player(session, 0);
        if (player == NULL) {
            DBG("====Debug player is null====");
            return;
        } else {
            DBG("====Debug player is not null====");
        }
    }

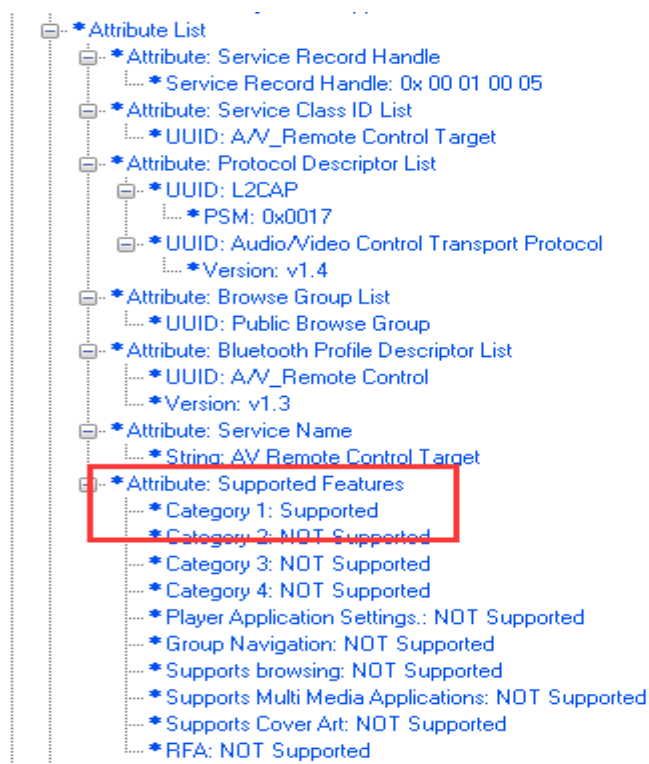
    if (controller->version < 0x0103)
        return;
}

```

```

profiles/audio/avrcp.c:controller_init() 0x2e36470 version 0x0104
profiles/audio/avrcp.c:controller_init() =====Debug player=====
profiles/audio/avrcp.c:controller_init() =====Debug player is not null=====
: profiles/audio/avrcp.c:controller_init() 0x2e3e410 version 0x0000

```



当时的接法办法：更换 MAC 地址后居然都没有问题了，将 8703 蓝牙 MAC

10: xx: 修改为 20:xx 但是根据 bluetoothd 的 LOG 查看代码，这个问题跟蓝牙 MAC 地址没有任何关系,结论:小米 6 蓝牙没有 iphone 支持完善，而是本身 8723DS 在整个 FW 工作机制出现了问题（怀疑跟上电时候 FW 状态不正常相关，整个可能解释了更换的 MAC 地址，问题机型不在复现到问题，这个问题先挂起，继续观察，概率低）

问题 2: 连接异常情况 2: 手机提示连接失败检查是否在通讯范围内

对方的 l2cap conn req 都被 8723ds 拒绝了,cfa 报文如下:

解决办法: 该类问题初期发现两类

A 类:某次开机, 蓝牙电源不正常, 做 reset 复位 开机启动脚本执行 hconfig hci0 reset

B 类: gatt 配网服务后, bluetoothd 被关闭了, 系统没有及时启动 bluetoothd,当连接的时候, 蓝牙底层无法通讯

| LT | Addr | CID | PSM | ID | Code | Source CID | Dest CID | Length | Fram |
|----|--------|-------------|-------|----|----------------------|------------|----------|--------|------|
| 4 | 0x0001 | (Signaling) | | 2 | Information request | | | 2 | |
| 4 | 0x0001 | (Signaling) | | 1 | Information request | | | 2 | |
| 4 | 0x0001 | (Signaling) | | 2 | Information response | | | 8 | |
| 4 | 0x0001 | (Signaling) | | 1 | Information response | | | 8 | |
| 4 | 0x0001 | (Signaling) | | 2 | Information request | | | 2 | |
| 4 | 0x0001 | (Signaling) | | 3 | Information request | | | 2 | |
| 4 | 0x0001 | (Signaling) | | 3 | Information response | | | 12 | |
| 4 | 0x0001 | (Signaling) | | 2 | Information response | | | 12 | |
| 4 | 0x0001 | (Signaling) | AVDTP | 4 | Connection request | 0x0041 | 0x0000 | 4 | |
| 4 | 0x0001 | (Signaling) | | 4 | Connection response | 0x0041 | 0x0000 | 8 | |
| 8 | 0x0001 | (Signaling) | | 3 | Information request | | | 2 | |
| 8 | 0x0001 | (Signaling) | | 1 | Information request | | | 2 | |
| 8 | 0x0001 | (Signaling) | | 2 | Information response | | | 8 | |
| 8 | 0x0001 | (Signaling) | | 1 | Information response | | | 8 | |
| 8 | 0x0001 | (Signaling) | | 2 | Information request | | | 2 | |
| 8 | 0x0001 | (Signaling) | | 3 | Information request | | | 2 | |
| 8 | 0x0001 | (Signaling) | | 3 | Information response | | | 12 | |
| 8 | 0x0001 | (Signaling) | | 2 | Information response | | | 12 | |
| 8 | 0x0001 | (Signaling) | SDP | 4 | Connection request | 0x0042 | 0x0000 | 4 | |
| 8 | 0x0001 | (Signaling) | | 4 | Connection response | 0x0042 | 0x0000 | 8 | |
| 8 | 0x0001 | (Signaling) | SDP | 5 | Connection request | 0x0043 | 0x0000 | 4 | |
| 8 | 0x0001 | (Signaling) | | 5 | Connection response | 0x0043 | 0x0000 | 8 | |

问题 3 异常断开连接

这类问题比较复杂, 有的是手机问题, 有的是蓝牙 FW 不稳定, 继续跟踪

问题 4 配对需要输入 PIN 码

读取 ssp mode: hciconfig hci0 sspmode 后, 居然是 disable

正常应该是 hci0: Type: Primary Bus: UART

BD Address: 0A:16:BA:CA:22:22 ACL MTU: 1021:8 SCO MTU: 255:12

Simple Pairing mode: Enabled

hciconfig hci0 sspmode 1

| Frame | Len | Type | Packet Type | Command | Response |
|-------|-----|------------|----------------|---------------------------|----------|
| 15 | 5 | Host | Command Packet | Write Simple Pairing Mode | |
| 16 | 7 | Controller | Event Packet | | Success |

引起问题原因: 蓝牙上电是默认 disable ssp 的, 但是协议栈没有将其 enable。

跟蓝牙开启脚本里面协议栈开启和上电时间有关, 目前优化完脚本未复现到该问题了

| Command | OCF | Command Parameters | Return Parameters |
|-------------------------------|--------|---------------------|-------------------|
| HCI_Write_Simple_Pairing_Mode | 0x0056 | Simple_Pairing_Mode | Status |

这个在内核代码里面，蓝牙协议栈部分代码是在内核里面的 net/bluetooth/hci_core.c

```

if (!imp_ssp_capable(hdev)) {
    /* When SSP is available, then the host features page
     * should also be available as well. However some
     * controllers list the max_page as 0 as long as SSP
     * has not been enabled. To achieve proper debugging
     * output, force the minimum max_page to 1 at least.
     */
    hdev->max_page = 0x01;

    if (hci_dev_test_flag(hdev, HCI_SSP_ENABLED)) {
        u8 mode = 0x01;

        hci_req_add(req, HCI_OP_WRITE_SSP_MODE,
                    sizeof(mode), &mode);
    } else {
        struct hci_cp_write_eir cp;

        memset(hdev->eir, 0, sizeof(hdev->eir));
        memset(&cp, 0, sizeof(cp));

        hci_req_add(req, HCI_OP_WRITE_EIR, sizeof(cp), &cp);
    }
}

```

如是典型的场景，不排除还有其他场景，暂时没发现到。

问题 5 扫描时间较长

有的手机能扫描到，**iphone 取消配对，忘记设备后**，注意（不是仅仅手机断开连接）iphone 要很长时间才能扫描到

查看当前 iscan 的参数：hciconfig hci0 inqparms



- 扫描窗口和扫描间隔设置的时间不能大于10.24S。
- 扫描窗口设置的值不能大于扫描间隔的值。
- 如果扫描窗口 = 扫描间隔的话，说明主机一直在进行扫描。

当前的 inq interval 2.56S

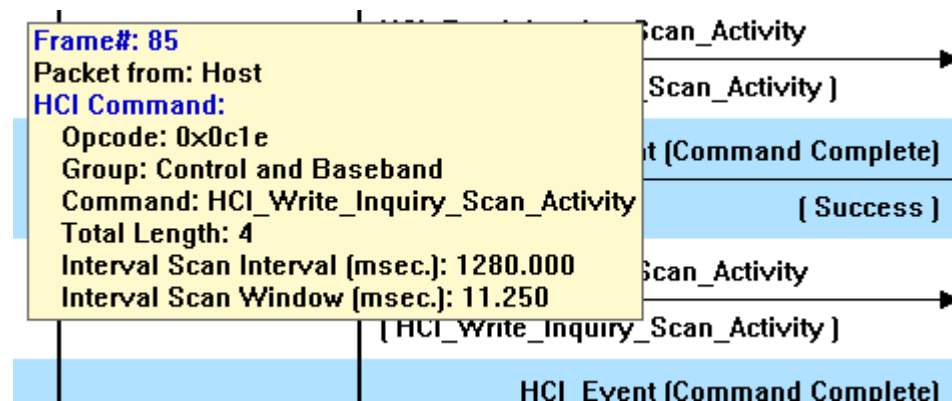
Inquiry interval: 4096 slots (2560.00 ms), window: 18 slots (11.25 ms)

Inquiry interval:两个连续的扫描窗口的起始时间之间的时间差，包括扫描休息的时间和扫描进行的时间

扫描窗口(scan window): 一次扫描进行的时间宽度

```
/ # hciconfig hci0 inqparms
hci0:   Type: Primary   Bus: UART
        BD Address: 00:2C:B0:00:22:22  ACL MTU: 1021:8  SCO MTU: 255:12
        Inquiry interval: 4096 slots (2560.00 ms), window: 18 slots (11.25 ms)
/ #
```

hciconfig hci0 inqparms 18: 2048, 调小 inq interval



相当于每 1.28s scan 一下 inquiry, inquiry 代表 iPhone 或者其他设备扫描 8723 的动作。

```
# hciconfig hci0 inqparms 18:2048
# hciconfig hci0 inqparms
hci0:   Type: Primary   Bus: UART
        BD Address: 00:2C:B0:00:22:22  ACL MTU: 1021:8  SCO MTU: 255:12
        Inquiry interval: 2048 slots (1280.00 ms), window: 18 slots (11.25 ms)
#
```

效果感觉好多了，但是这个修改没有导入项目，这个值对蓝牙 FW 是否有影响需要测试。

5. reset 后, class 变了

update_class 看起来是内核去做了 update, 尝试了如下修改, 但是没用, 反而导致蓝牙异常

diff --git a/net/bluetooth/mgmt.c b/net/bluetooth/mgmt.c

index b1b0a1c0bd8d..fb635714e05a 100644

--- a/net/bluetooth/mgmt.c

+++ b/net/bluetooth/mgmt.c

@@ -7663,7 +7663,7 @@ static int powered_update_hci(struct hci_dev *hdev)

else

write_fast_connectable(&req, false);

__hci_update_page_scan(&req);

- update_class(&req);

+ //update_class(&req);

update_name(&req);

update_eir(&req)

解决办法：up 之后才去设置 class

6 . 关于 BLE 蓝牙连接一些参数 (Supervision Timeout slaveLatency connectionInterval)

这些参数再 core 文档里面如下章节有描述，可以通过 hci cmd 下发

7.8.18 LE Connection Update Command

| Command | OCF | Command parameters | Return Parameters |
|-------------------------------|--------|--|-------------------|
| HCI_LE_Connec- tion_Update | 0x0013 | Connection_Handle, Conn_Interval_Min, Conn_Interval_Max, Conn_Latency, Supervision_Timeout, Minimum_CE_Length, Maximum_CE_Length | |

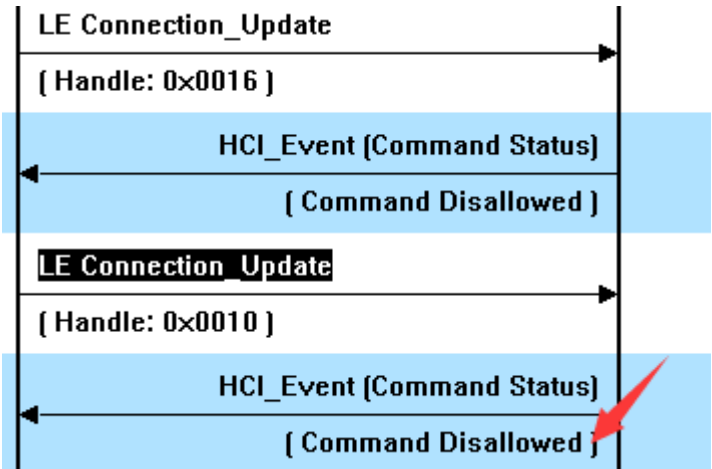
Description:

例子：hcitool -i hci0 cmd 0x08 0x0013 00 00 00 00 00 05 0c 80 01 f3 0c 80
00 00 00 00 具体参数规范只能参考协议里面的值进行修改。或者用 hcitool lecup 命令

```
/ # hcitool lecup
lecup: too few arguments (minimal: 5)
Usage:
    lecup <handle> <min> <max> <latency> <timeout>
Options:
    --handle=<0xFFFF>  LE connection handle
    --min=<interval>    Range: 0x0006 to 0x0C80
    --max=<interval>    Range: 0x0006 to 0x0C80
    --latency=<range>   Slave latency. Range: 0x0000 to 0x03E8
    --timeout=<time>    N * 10ms. Range: 0x000A to 0x0C80

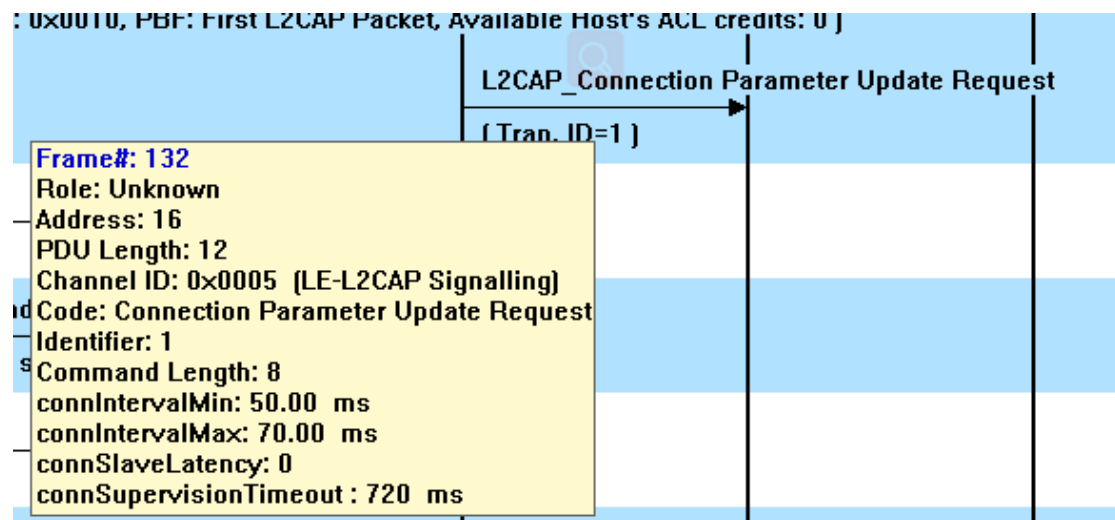
    min/max range: 7.5ms to 4s. Multiply factor: 1.25ms
    timeout range: 100ms to 32.0s. Larger than max interval
```

或者用 hcitool -i hci0 lecup 16 0x0006 0x0c80 0x0001 0x0c80



设置下去后，抓包发现手机不支持，显示 disallowed，但是在连接 BLE 后，也发现 le meta conn

complete event 里有这些参数，以上命令只有在 BLE 建立连接后才有意义。



7. 长时间播放 A2DP SINK 音频卡顿问题，比如可能出现 1 小时卡顿 2 次到 3 次，该问题小雅暂时接受，这个问题应该作为专项继续研究，在 RK3308_WIFI_bt_共存问题排查.pdf 文档中有一些测试和数据结果。

```
331.076622] rtk_btcoex: count_a2dp_packet_timeout: a2dp_packet_count 63
luealsa-aptplay: aptplay.c:391: An underrun has occurred
332.078616] rtk_btcoex: count_a2dp_packet_timeout: a2dp_packet_count 77
333.080655] rtk_btcoex: count_a2dp_packet_timeout: a2dp_packet_count 70
334.082670] rtk_btcoex: count_a2dp_packet_timeout: a2dp_packet_count 74
luealsa-aptplay: aptplay.c:391: An underrun has occurred
luealsa-aptplay: aptplay.c:391: An underrun has occurred
luealsa-aptplay: aptplay.c:391: An underrun has occurred
```

```
snd_pcm_uframes_t frames = ((buffer_tail - buffer) + 1) / 1;
if ((err = snd_pcm_writei(w->pcm, buffer, frames)) < 0)
    switch (-err) {
        case EPIPE:
            debug("An underrun has occurred");
            snd_pcm_prepare(w->pcm);
            break;
    }
```

Returns:

a positive number of frames actually written otherwise a negative error code

Return values:

| | |
|-----------|---|
| -EBADFD | PCM is not in the right state (<u>SND_PCM_STATE_PREPARED</u> or <u>SND_PCM_STATE_RUNNING</u>) |
| -EPIPE | an underrun occurred |
| -ESTRPIPE | a suspend event occurred (stream is suspended and waiting for an application recovery) |

结论：蓝牙收到的数据不够了，导致往下写的不够用了
目前处理状态：

- 1. 原厂更新 FW 和驱动工作机制（有一定改善，但是还是存在一些问题，是干扰和共存之间没有找到一个最好的平衡点）
- 2. 我们这个 PCM 声卡处理数据引入 FIFO 机制，（实际测试效果不明显，感觉没用）

6. BLE 广播问题，概率会出现如下问题：

此问题经过与 APK 方面讨论，是 APP 端 IOS 和 android 系统的读取 characteristic 写入有 BUG，

- 1. APK 找不到 WIFI_SERVICE_UUID 已经解决
- 2. APK 发送过来的 SSID PASSD 为空 已经解决

7. 不同型号手机兼容性探究

主要表现：

- 1：有的手机连接蓝牙后，AVRCP 状态不正常，
 - 2：有的手机连接不上蓝牙
- 以上两种 CASE，不正常的手机连接不上，换个手机又没有问题，典型的现象手机重启后也恢复正常。

8： 蓝牙底层发起断开连接请求

底层蓝牙链路产生的超时断线

```
#define MGMT_DEV_DISCONN_TIMEOUT 0x01
```

```

bluetoothd[24514]: profiles/audio/player.c:media_player_set_status() stopped
bluetoothd[24514]: src/adapter.c:dev disconnected() Device D8:1D:72:1C:26:DC disconnected, reason 1
bluetoothd[24514]: src/adapter.c:adapter_remove_connection()
bluetoothd[24514]: plugins/policy.c:disconnect_cb() reason 1
bluetoothd[24514]: plugins/policy.c:disconnect_cb() Device /org/bluez/hci0/dev_D8_1D_72_1C_26_DC identified for auto-reconnection
bluetoothd[24514]: plugins/policy.c:reconnect_set_timer() attempt 1/7 1 seconds
bluetoothd[24514]: src/adapter.c:bonding_attempt_complete() hci0 bdaddr D8:1D:72:1C:26:DC type 0 status 0xe
bluetoothd[24514]: src/device.c:device_bonding_complete() bonding (nil) status 0x0e
bluetoothd[24514]: src/device.c:device_bonding_failed() status 14

```

目前测试先关闭 **WIFI**，继续拷机，确认是否是共存引起的蓝牙超时。目前还在拷机测试中，

蓝牙语音：

1. 目前 RK3308 采用 bluez-alsa 进行 A2DP sink 音频播放，手机端调大 调小 音量 bluez-alsa 行为不一样。

2. 将 bluez-alsa 最终处理 scale 数据在 io_thread_scale_pcm scal 函数添加

分两个方法测试：

1. 注释掉 _a2dp_sink_sbc scale_pcm 处理

2. 将默认音量修改，默认是 127

```

void snd_pcm_scale_s16le(int16_t *buffer, size_t size, int
channels,

    double ch1_scale, double ch2_scale) {
    switch (channels) {
    case 1:
        if (ch1_scale != 1.0)
            while (size--)
                buffer[size] = buffer[size] * ch1_scale;
        break;
    case 2:
        if (ch1_scale != 1.0 || ch2_scale != 1.0)
            while (size--) {
                double scale = size % 2 == 0 ? ch1_scale :
ch2_scale;

```

```

        buffer[size] = buffer[size] * scale;

    }

    break;

}

```

```

}

```

```

diff --git a/src/bluez.c b/src/bluez.c

old mode 100644
new mode 100755
index 78e2093..9fa00f9
--- a/src/bluez.c
+++ b/src/bluez.c

@@ -288,7 +288,7 @@ static int
bluez_endpoint_set_configuration(GDBusMethodInvocation
*inv, void *us

    const char *transport;

    char *device = NULL, *state = NULL;

    uint8_t *configuration = NULL;

-    uint16_t volume = 127;
+    uint16_t volume = 90;

    uint16_t delay = 150;

    size_t size = 0;

    int ret = 0;

@@ -506,6 +506,7 @@ static int
bluez_endpoint_set_configuration(GDBusMethodInvocation
*inv, void *us

```

```

/* received volume is in range [0,
127]*/

volume = g_variant_get_uint16(value);

+         debug("key -> Volum
bluez_endpoint_set_configuration volume is  %d",volume);

}

```

```

@@ -538,7 +539,7 @@ static int
bluez_endpoint_set_configuration(GDBusMethodInvocation
*inv, void *us

        bluetooth_profile_to_string(profile,
codec), batostr_(&d->addr));

        debug("Configuration: channels: %u, sampling: %u",

                transport_get_channels(t),
transport_get_sampling(t));

-
+   debug("transport_new_a2dp set volume is  %d",volume);

        transport_set_state_from_string(t, state);

-
+   debug("transport_new_a2dp set volume is  %d",volume);

        transport_set_state_from_string(t, state);

        g_dbus_method_invocation_return_value(inv, NULL);

diff --git a/src/ctl.c b/src/ctl.c

old mode 100644

new mode 100755

```

```

diff --git a/src/io.c b/src/io.c
old mode 100644
new mode 100755
index 2744677..e5d0274
--- a/src/io.c
+++ b/src/io.c
@@ -281,7 +281,8 @@ void *io_thread_a2dp_sink_sbc(void *arg)
{
    }

    const size_t samples = output - out_buffer;
-    io_thread_scale_pcm(t, out_buffer, samples,
channels);
+    /*add by rockchip-xxh debug for bluealse
scale*/
+    //io_thread_scale_pcm(t, out_buffer,
samples, channels);
    if (io_thread_write_pcm(&t->a2dp.pcm,
out_buffer, samples) == -1)
        error("FIFO write error: %s",
strerror(errno));

```

实际测试结果:

1. android 手机, 手机端直接改变了 media data 的音量, bluez-alsa 不做处理, 从打印结果来看, 1:1 传输,
2. IOS 手机, 手机并不会改变 media data 音量, 而是发送 avrcp set absolute volume 给 speaker, 所以在 iOS 上可以看到一个带蓝牙标志的音量条在改变, 打印结果, bluealsa 会收到 org.bluez.MediaTransport1.Volume 消息, 从而改变 media data 数据。
3. 所以如果注释掉 scale 代码, 在 iOS 上改变音量条实际应该不会改变音量, 个人觉得最好的做法是在收

到 org.bluez.MediaTransport1.Volume 消息时通过 alsamixer 改变本地音量，当然现在 bluealsa 框架是不会调用 alsamixer 调节音量的，它直接改变(scale)收到的 media data 而已

4. amixer -D bluealsa sset 'xxh - A2DP' 100% //这个命令的意思，通过 bluealsa_write_integer 直接改变 sink 端音量，此函数调用 bluealsa_set_transport_volume() 给 bluealsa 发消息 io_thread_scale_pcm 也就是改变本地 sink 端的音量；一句话，也就是 amixer 也可以改变 sink 收到的音频数据。

5. sink 不一定能控制远端的音量的，这个要看手机是否支持 avrcp set abs volume 命令，(AVRCP_SET_ABSOLUTE_VOLUME)

doc/media-api.txt

```
MediaEndpoint1 hierarchy
=====
Service      unique name
Interface    org.bluez.MediaEndpoint1
Object path  freely definable
Methods      void SetConfiguration(object transport, dict properties)
              Set configuration for the transport.
              array{byte} SelectConfiguration(array{byte} capabilities)
              Select preferable configuration from the supported
              capabilities.
              Returns a configuration which can be used to setup
              a transport.
              Note: There is no need to cache the selected
              configuration since on success the configuration is
              send back as parameter of SetConfiguration.
              void ClearConfiguration(object transport)
              Clear transport configuration.
              void Release()
              This method gets called when the service daemon
              unregisters the endpoint. An endpoint can use it to do
              cleanup tasks. There is no need to unregister the
              endpoint, because when this method gets called it has
              already been unregistered.

              acquired by the sender.
              uint16 volume [readwrite]
              Optional. Indicates volume level of the transport,
              this property is only writeable when the transport was
              acquired by the sender.
              Possible values: 0-127

static const GDBusPropertyTable transport_properties[] = {
    {"Device", "o", get_device },
    {"UUID", "s", get_uuid },
    {"Codec", "y", get_codec },
    {"Configuration", "ay", get_configuration },
    {"State", "s", get_state },
    {"Delay", "q", get_delay, NULL, delay_exists },
    {"Volume", "q", get_volume, set_volume, volume_exists },
};
```

#define MEDIA_TRANSPORT_INTERFACE "org.bluez.MediaTransport1"

见 profiles/audio/avrcp.c avrcp_set_volume 函数设置 source 端音量

```
int avrcp_set_volume(struct btd_device *dev, uint8_t volume, bool notify)
{
    struct avrcp_server *server;
    struct avrcp *session;
    uint8_t buf[AVRCP_HEADER_LENGTH + 1];
    struct avrcp_header *pdu = (void *) buf;

    server = find_server(servers, device_get_adapter(dev));
    if (server == NULL)
        return -EINVAL;

    session = find_session(server->sessions, dev);
    if (session == NULL)
        return -ENOTCONN;

    if (notify) {
        if (!session->target)
            return -NOTSUP;
        return avrcp_event(session, AVRCP_EVENT_VOLUME_CHANGED,
                           &volume);
    }

    if (!session->controller || session->controller->version < 0x0104)
        return -NOTSUP;

    memset(buf, 0, sizeof(buf));
    set_company_id(pdu->company_id, IEEEID_BT_SIG);

    pdu->pdu_id = AVRCP_SET_ABSOLUTE_VOLUME;
    pdu->params[0] = volume;
    pdu->params_len = htons(1);

    return avctp_send_vendordep_req(session->conn,
                                     AVC_CTYPE_CONTROL, AVC_SUBUNIT_PANEL,
                                     buf, sizeof(buf),
                                     avrcp_handle_set_volume, session);
}
```

dbus-send --print-reply --system --dest=org.bluez /org/bluez/hci0/dev_D8_1D_72_1C_26_DC/fd0
org.freedesktop.DBus.Properties.GetAll string:"org.bluez.MediaTransport1"

查看所有属性，可以看到 Volum 属性，那么就可以使用 set 方法去设置这个属性了

```
/ # dbus-send --print-reply --system --dest=org.bluez /org/bluez/hci0/dev_D8_1D_72_1C_26_DC/fd0 org.freedesktop.DBus.Properties.GetAll string:"org.bluez.MediaTransport1"
method return time=1548747170.664427 sender=:1.1 -> destination=:1.45 serial=140 reply_serial=2
array [
  dict entry(
    string "Device"
    variant
      object path "/org/bluez/hci0/dev_D8_1D_72_1C_26_DC"
  )
  dict entry(
    string "UUID"
    variant
      string "0000110B-0000-1000-8000-00805F9B34FB"
  )
  dict entry(
    string "Codec"
    variant
      byte 0
  )
  dict entry(
    string "Configuration"
    variant
      array of bytes [
        21 15 02 35
      ]
  )
  dict entry(
    string "State"
    variant
      string "active"
  )
  dict entry(
    string "Volume"
    variant
      uint16 127
  )
]
```

命令如下：

查看：**org.bluez.MediaTransport1.Volume**

dbus-send --print-reply --system --dest=org.bluez /org/bluez/hci0/dev_D8_1D_72_1C_26_DC/fd0
org.freedesktop.DBus.Properties.Get string:"org.bluez.MediaTransport1" "string:Volume" | cut -d"
" -f 12

设置 org.bluez.MediaTransport1.Volume

```
dbus-send --print-reply --system --dest=org.bluez /org/bluez/hci0/dev_D8_1D_72_1C_26_DC/fd0
org.freedesktop.DBus.Properties.Set          string:org.bluez.MediaTransport1
variant:uint16:90
```

```
/ #
/ # dbus-send --print-reply --system --dest=org.bluez /org/bluez/hci0/dev_D8_1D_
72_1C_26_DC/fd0 org.freedesktop.DBus.Properties.Set string:org.bluez.MediaTransp
ort1 string:Volume variant:uint16:90 ← 设置90
method return time=1548747503.412914 sender=:1.1 -> destination=:1.50 serial=148 reply_serial=2
/ # dbus-send --print-reply --system --dest=org.bluez /org/bluez/hci0/dev_D8_1D_
72_1C_26_DC/fd0 org.freedesktop.DBus.Properties.Get string:"org.bluez.MediaTrans
port1" "string:Volume" | cut -d" " -f 12

90 ← 查看Volum成功
/ # dbus-send --print-reply --system --dest=org.bluez /org/bluez/hci0/dev_D8_1D_
72_1C_26_DC/fd0 org.freedesktop.DBus.Properties.Set string:org.bluez.MediaTransp
ort1 string:Volume variant:uint16:50
method return time=1548747522.961710 sender=:1.1 -> destination=:1.52 serial=151 reply_serial=2
/ # dbus-send --print-reply --system --dest=org.bluez /org/bluez/hci0/dev_D8_1D_
72_1C_26_DC/fd0 org.freedesktop.DBus.Properties.Get string:"org.bluez.MediaTrans
port1" "string:Volume" | cut -d" " -f 12
```