Ford Motor (China) Company

蓝牙电话结构流程分析

Version <1.0>

免责申明

本文档中的内容仅供参考。福特汽车中国对本服务内容的错误或遗漏概不负责。在任何情况下，福特汽车中国均不对因使用本文档而产生或与之相关的任何特殊，直接，间接，间接或偶然的损害赔偿或任何损害负责，无论是在合同，疏忽，其他侵权行为中服务或服务的内容。福特汽车中国保留随时对本文档内容进行补充，删除或修改的权利，恕不另行通知。

Disclaimer

The contents contained in this document are for general information purposes only. Ford Motor China assumes no responsibility for errors or omissions in the contents on the Service.

In no event shall Ford Motor China be liable for any special, direct, indirect, consequential, or incidental damages or any damages whatsoever, whether in an action of contract, negligence or other tort, arising out of or in connection with the use of the Document or the contents of the Document.

Ford Motor China reserves the right to make additions, deletions, or modification to the contents on the Service at any time without prior notice.

编制	Beyondsoft	日期	2017-12-7	版权	署名-相同方式共享 4.0 国际
审核	Ford	日期	2017-12-7	管理	Ford

修改历史

序号	日期	说明
1.0.0	2017-11-25	完成总体架构，以及 AT 指令部分
1.0.1	2017-12-3	添加 AT 指令（ATD、AT+BLDN）、交互流程（服务级连接、注册状态、音频连接、拨打电话），以及 nohands 技术调研
1.0.2	2017-12-8	加入描述、蓝牙协议栈、蓝牙框架结构、HCI 协议分析。并调整了文档的部分结构。

目　　录

1 目的

本文档主要对蓝牙电话整体结构进行分析，包括主体架构、协议栈、主体流程、以及协议说明。为下一步的开发提供帮助。

2 描述

蓝牙按照蓝牙协议的逻辑功能，协议堆栈由下至上分为三个部分：传输协议、中介协议和应用协议。简介如下：

- 传输协议负责蓝牙设备间相互确认对方的位置，以及建立和管理蓝牙设备间的物理和逻辑链路。这一部分又分为低层传输协议和高层传输协议两部分。低层传输协议包括蓝牙的射频（Radio）部分、基带与链路控制器（Baseband & Link Controller）和链路管理器协议（Link Manager Protocol，LMP）. 低层传输协议侧重于语音与数据无线传输的物理实现以及蓝牙设备间的连接与组网。高层传输协议包括逻辑链路控制与适配协议（Logical Link Control and Adaptation Protocol，L2CAP）和主机控制器接口（Host Controller Interface，HCI）。HCI 公为应用协议堆栈的高层部分提供了一个访问低层传输协议的指令接口。
- 中介协议层为高层应用协议或程序在蓝牙逻辑链路上工作提供了必要的支持，为应用层提供了各种不同的标准接口。这部分协议包括以下几个部分：串口仿真协议（RFCOMM）、服务发现协议（Service Discovery Protocol，SDP）、IrDA（Infrared Data Association，红外数据协会）、网络访问协议、电话控制协议。

  电话控制协议包括 TCS、AT 指令集和音频。电话控制协议（Telephone Control Protocol Specification，TCS）是基于国际电信联盟电信组（International Telecommunication Union-Telecommunication，ITU-T）的 Q.931 标准制定的，用于支持电话功能，用于实现多用户模式下对移动电话和调制解调器的控制。蓝牙直接在基带上处理音频信号（主要指数字语音信号），采用 SCO 链路传输语音。
- 应用协议：是指那些位于蓝牙协议堆栈之上的应用协议和其中所涉及的协议。蓝牙规范提供了传输层及中介层定义和应用框架，在传输层及中介层之上，不同的蓝牙设备必须采用统一符合蓝牙规范的形式；而在应用层上，完全由开发人员自主实现。

蓝牙免提框架（HFP）中，音频网关（Audio Gateway，AG）是指音频输入输出网关设备，典型的音频网关就是蜂窝移动电话。免提单元（HandFree，HF）是指可以遥控音频网关的设备，如嵌入汽车内的免提耳麦。

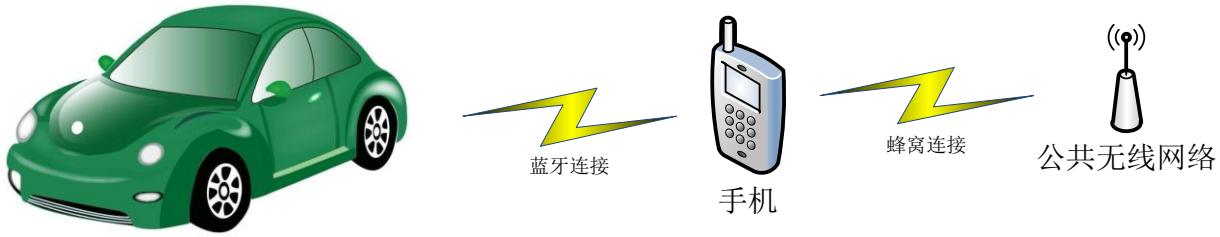
3 缩写术语

术语	全称	描述
HCI	Host Control Interface	主机控制接口

L2CAP	Logical Link Control and Adaptation Protocol	逻辑链路控制和适配协议
SDP	Service Discovery Protocol	服务发现协议
RFCOMM		基于 ETSI TS 7.10 串行电缆仿真协议，即蓝牙串口仿真协议
SCO	Synchronous Connection Oriented	同步定向连接方式，主要用于对时间要求很高的数据通信，如语音等。
SPP	Serial Port profile	蓝牙串口访问协议
HFP	Hands-Free Profile	蓝牙免提协议
GAP	Generic Access Profile	通用访问配置协议
LMP	Link Management Protocol	链路管理协议
AG	Audio Gateway	手机端音频网关
HF	Hands-Free	主机蓝牙免提
A2DP	Advanced Audio Distribution Profile	高级音频传输协议
ACL	Asynchronous Connectionless	异步无连接方式，主要用于对时间要求不敏感的数据通信，如文件数据或控制指令等
OGF	OpCode Group Field	指令分组格式中 OpCode 组字段
OCF	OpCode Command Field	指令分组格式中 OpCode 命令字段

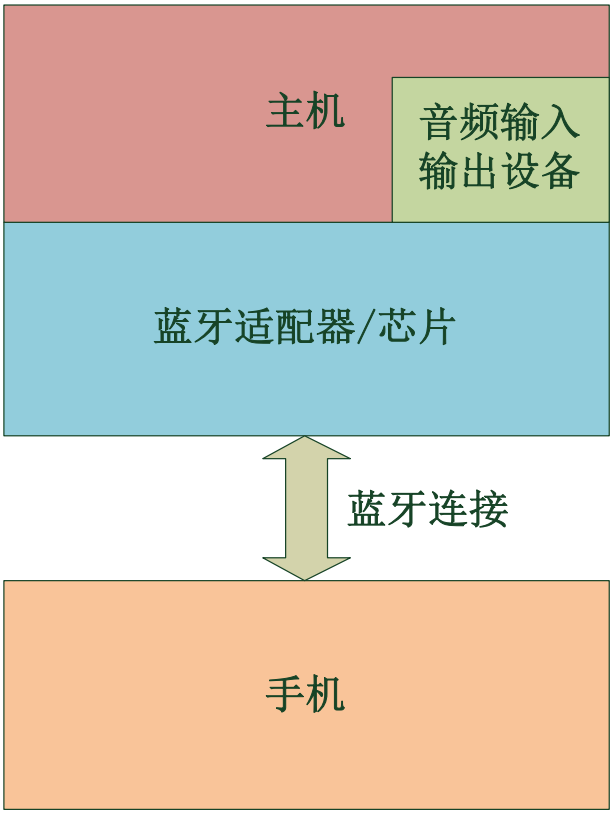
4 物理结构

4.1 HFP 网络拓扑图



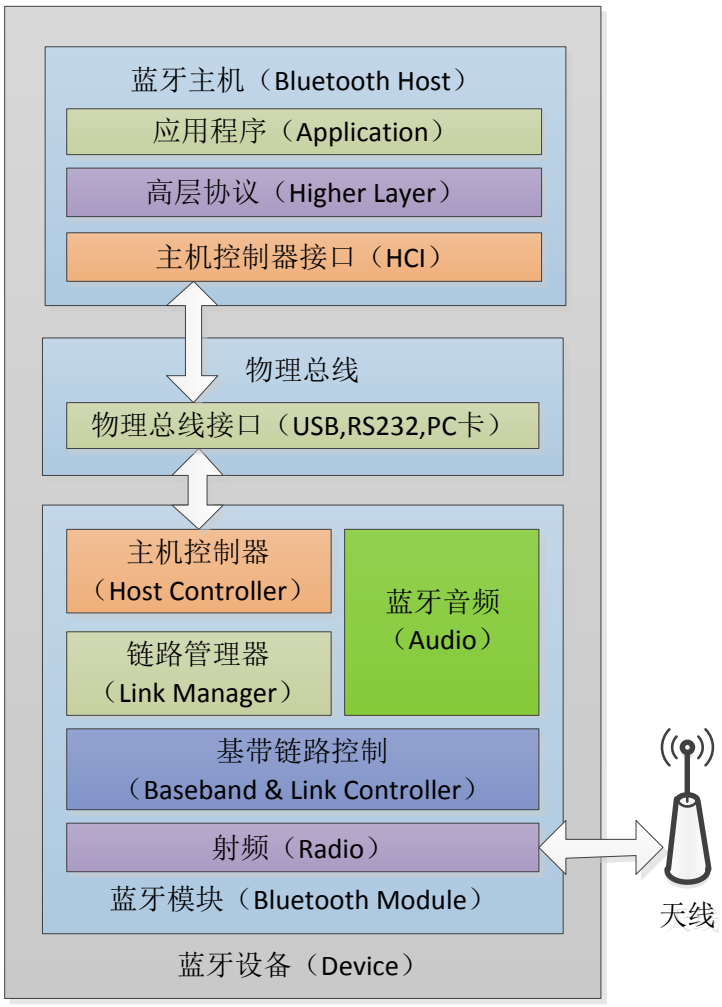
车载HF设备

4.2 HFP 物理层次图



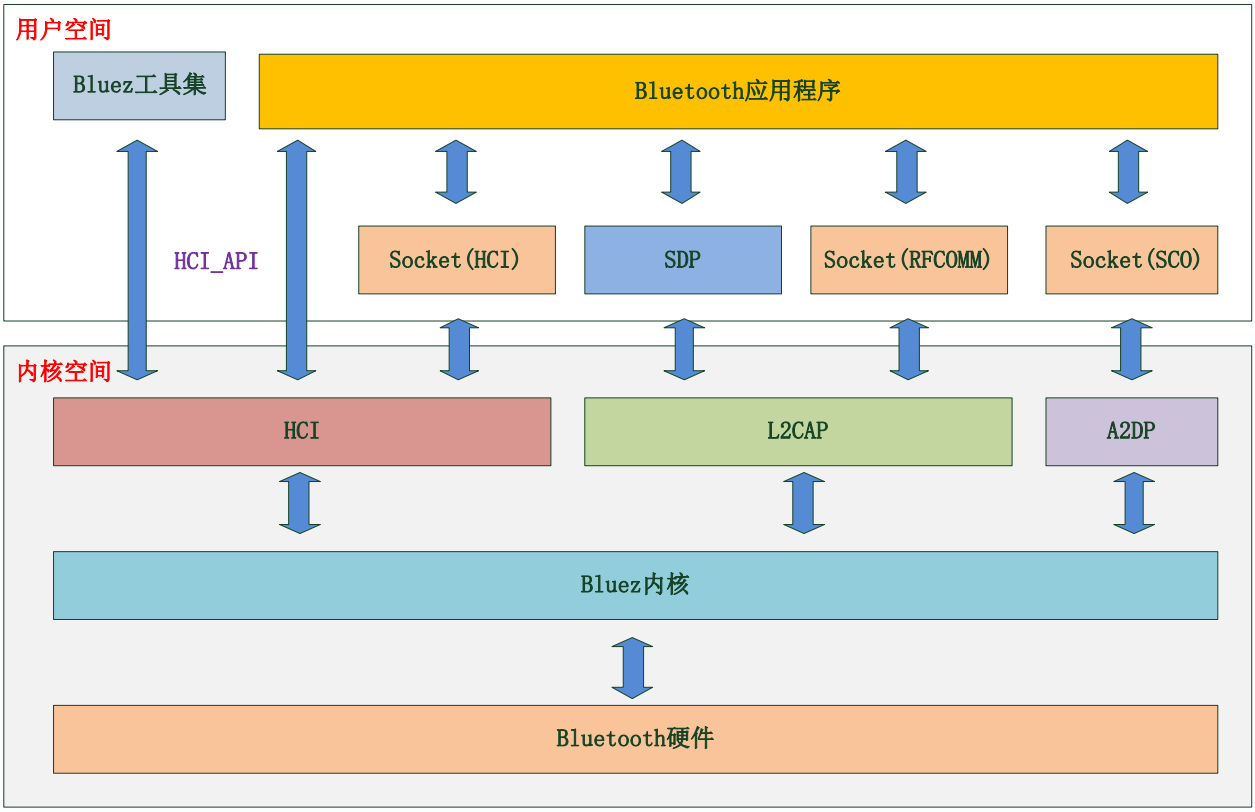
5 逻辑结构

5.1 蓝牙协议栈



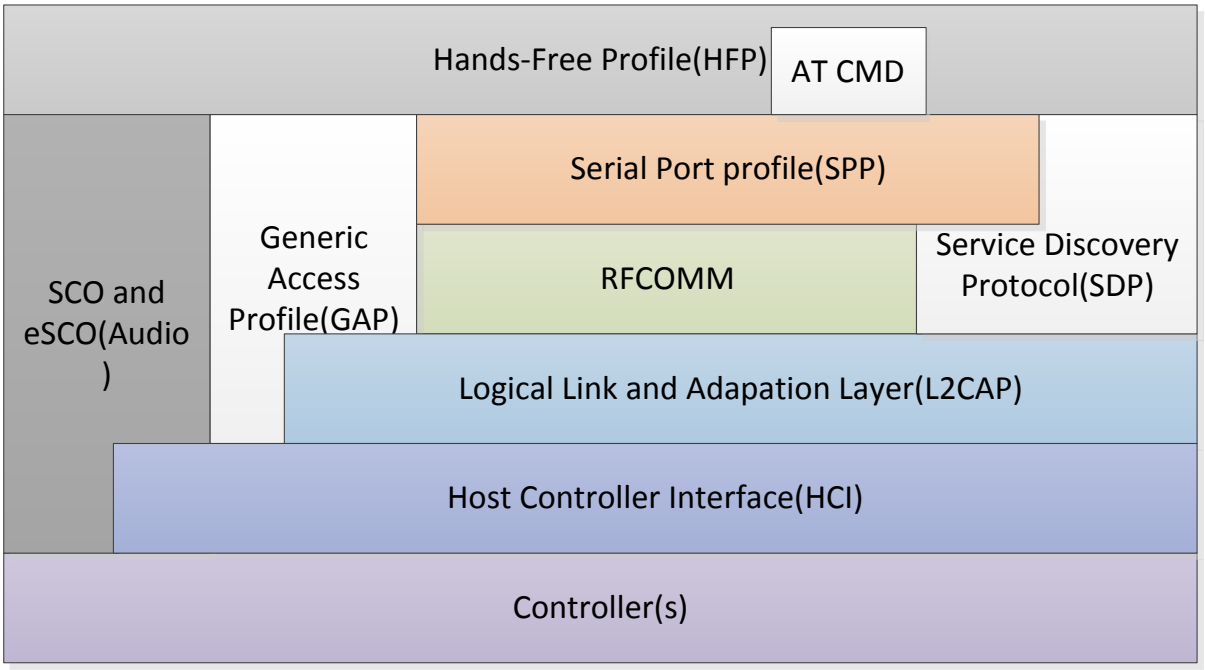
层次名称	说明
射频（Radio）	载波产生、信号调制、数据收发、功率控制、信号强度
基带与链路控制器（Baseband & Link Controller）	跳频选择、蓝牙编址、链路类型、信道编码、收发规则、信道控制、音频规范、安全设置。包括物理链接 ACL 和 SCO
链路管理器（Link Manager）	对本地或远端蓝牙设备的链路性能进行设置和管理，包括：链路管理、安全管理、功率管理、质量管理、传输调度
主机控制器（Host Controller）	提供了一个控制基带与链路控制器、链路管理器等硬件的统一接口。HCI 分组有三种类型：指令分组（Command Packet）、事件分组（Event Packet）和数据分组（Data Packet）。数据分组分为异步（ACL）数据和面向连接（SCO）的同步数据分组。HCI 的指令分组类型包括链路控制指令、链路策略与模式指令、主机控制与基带指令、信息指令、状态指令和测试指令。
主机控制器接口（Host Controller Interface）	
高层协议	包括 GAP，是其它所有应用框架协议的基础。

5.2 蓝牙框架结构



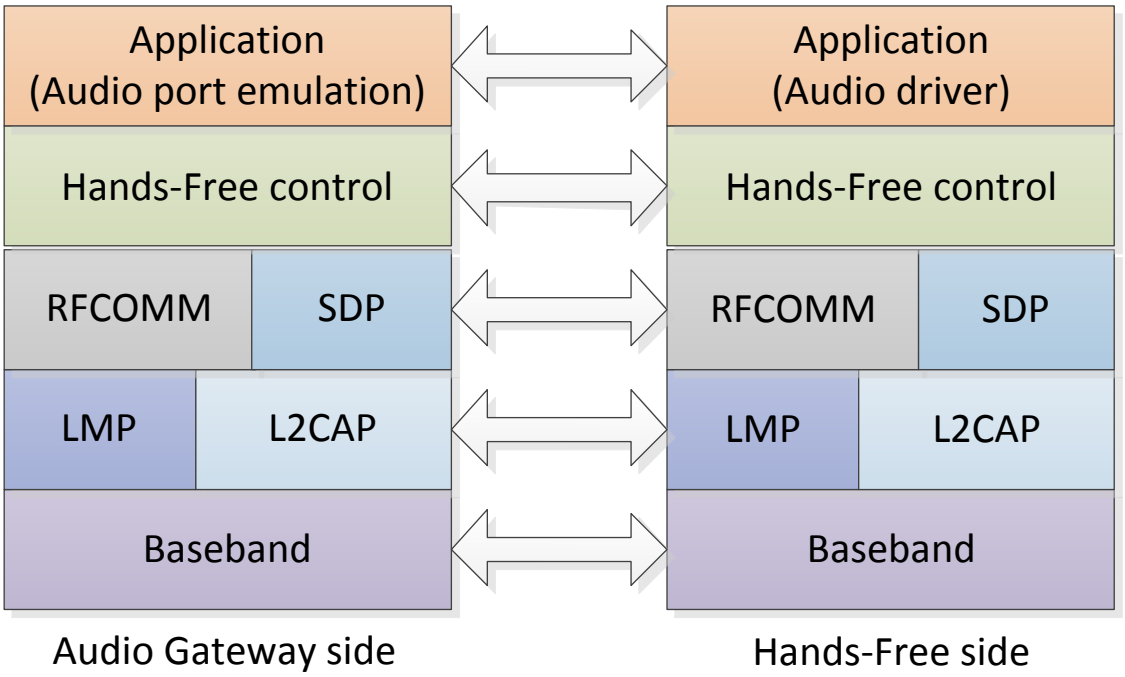
模块	描述
Bluetooth 应用程序	指需要使用蓝牙适配器与手机或其它蓝牙终端进行通信的应用程序
Bluez 工具集	指 Bluez 提供的可以操作 HCI 的一系列接口
Socket(HCI)	指 Bluetooth 应用程序可以通过 socket 与 HCI 进行通信
SDP	指服务发现协议，用户可以连接指定的 UUID 的蓝牙服务，并返回 channel
Socket(RFCOMM)	指 Bluetooth 应用程序可以通过 socket，连接蓝牙的指定 channel，并与之通信
Socket(SCO)	指 Bluetooth 应用程序可以通过 socket，连接蓝牙音频，发送与接收音频数据
HCI	指主机控制接口，包括查找、匹配蓝牙设备，获取设备名称等操作
L2CAP	指逻辑链路控制与适配协议，包括上层 SDP 和 RFCOMM 协议
A2DP	蓝牙音频传输模型协议，包括 SCO 协议
Bluez 内核	实现各上层协议，并转换为指令，通过硬件接口下发到 Bluetooth 硬件
Bluetooth 硬件	指蓝牙适配器、或蓝牙芯片，包括 CRS8510 等

5.3 HFP 逻辑层次结构



层次名称	说明
Controller(s)	HCI 驱动单元及硬件控制单元
Host Controller Interface(HCI)	HCI 接口单元，通过调用和控制 HCI 驱动单元，来控制蓝牙适配单元
Logical Link and Adapation Layer(L2CAP)	逻辑链路控制和适配协议，是蓝牙系统中的核心协议，负责适配基带中的上层协议
RFCOMM	为 L2CAP 提供上层的串口仿真协议
Service Discovery Protocol(SDP)	查找指定 channel 的蓝牙服务
Serial Port profile(SPP)	为 RFCOMM 和 SDP 提供一套上层的基于 socket 的访问接口
Hands-Free Profile(HFP)	蓝牙免提协议，包括 AT 指令集
SCO and eSCO(Audio)	面向连接的基于 socket 的音频传输方式
Generic Access Profile(GAP)	定义设备连接的相关服务

5.4 HFP 协议栈



层次名称	说明
Baseband	蓝牙基带
LMP	链路管理，用于建立 ACL、SCO 链路
Audio Gateway size	AG 端
Hands-Free side	HF 端

6 协议说明

6.1 HCI 协议分析

6.1.1 HCI 分组类型（HCI_Packet_Types）

分组类型	宏	值
指令分组	HCI_COMMAND_PKT	0x01
ACL 数据分组	HCI_ACLDATA_PKT	0x02
SCO 数据分组	HCI_SCODATA_PKT	0x03
事件分组	HCI_EVENT_PKT	0x04
错误消息分组		0x05
协商分组		0x06

6.1.2 操作码组类型（HCI 指令分组类型）（OGF）

操作码组类型	宏	值
链路控制指令	OGF_LINK_CTL	0x01
链路策略指令	OGF_LINK_POLICY	0x02
主机控制器与基带指令	OGF_HOST_CTL	0x03
信息指令参数	OGF_INFO_PARAM	0x04
状态指令参数	OGF_STATUS_PARAM	0x05
测试指令	OGF_TESTING_CMD	0x3e
错误代码		

6.1.3 操作码指令类型_链路控制指令（OCF）

操作码指令类型	值	描述
Inquiry	0x0001	蓝牙设备进入查询模式，搜索临近设备
Inquiry Cancel	0x0002	退出查询模式
Periodic Inquiry Mode	0x0003	蓝牙设备在指定周期内自动查询
Exit Periodic Inquiry Mode	0x0004	退出自动查询模式
Create Connection	0x0005	按指定蓝牙设备的 BD_ADDR 创建 ACL 链路
Disconnect	0x0006	终止现有连接
Add SCO Connection	0x0007	利用连接句柄参数指定的 ACL 连接创建 SCO
Cancel Create Connection	0x0008	
Accept Connection Request	0x0009	接收新的呼入连接请求
Reject Connection Request	0x000A	拒绝新的呼入连接请求
Link Key Request Reply	0x000B	应答从主机控制器发出的链路密钥请求事件，并指定存储在主机上的链路密钥做为与 BD_ADDR 指

		定的蓝牙设备进行连接使用的链路密钥请求事件
Link Key Request Negative Reply	0x000C	如果主机上没有存储链路密钥，作为与 BD_ADDR 指定的蓝牙设备进行连接使用的链路密钥，就应答从主机控制器发出的链路密钥请求事件
PIN Code Request Reply	0x000D	应答从主机控制器发出的 PIN 请求事件，并指定用于连接的 PIN
PIN Code Request Negative Reply	0x000E	当主机不能指定连接的 PIN 时，应回答从机控制器发出的 PIN 请求事件
Change Connection Packet Type	0x000F	改变正在建立连接的分组类型
Authentication Request	0x0011	指定连接句柄关联的两个蓝牙设备之间建立身份鉴权
Set Connection Encryption	0x0013	建立取消连接加密
Change Connection Link Key	0x0015	强制关联了连接句柄的两个设备建立连接，并生成一个新的链路密钥
Master Link Key	0x0017	强制关联了连接句柄的两个设备利用主设备时链路密钥或常规密钥
Remote Name Request	0x0019	获取远端设备的名称
Remote Name Request Cancel	0x001A	退出获取远端设备名称操作
Read Remote Supported Features	0x001B	请求远端设备所支持的特性列表

Read Remote Extended Features	0x001C	请求远端设备扩展的特性列表
Read Remote Version Information	0x001D	从远端设备读取版本信息
Read Clock Offset	0x001F	读取远端的时钟信息

6.1.4 事件分组类型（Event_Types）

事件分组类型	值	描述
EVT_INQUIRY_COMPLETE	0x01	
EVT_INQUIRY_RESULT	0x02	
EVT_CONN_COMPLETE	0x03	
EVT_CONN_REQUEST	0x04	
EVT_DISCONN_COMPLETE	0x05	
EVT_AUTH_COMPLETE	0x06	
EVT_REMOTE_NAME_REQ_COMPLETE	0x07	
EVT_ENCRYPT_CHANGE	0x08	
EVT_CHANGE_CONN_LINK_KEY_COMPLETE	0x09	
EVT_MASTER_LINK_KEY_COMPLETE	0x0A	
EVT_READ_REMOTE_FEATURES_COMPLETE	0x0B	
EVT_READ_REMOTE_VERSION_COMPLETE	0x0C	
EVT_QOS_SETUP_COMPLETE	0x0D	
EVT_CMD_COMPLETE	0x0E	

EVT_CMD_STATUS	0x0F	
EVT_HARDWARE_ERROR	0x10	
EVT_FLUSH_OCCURRED	0x11	
EVT_ROLE_CHANGE	0x12	
EVT_NUM_COMP_PKTS	0x13	
EVT_MODE_CHANGE	0x14	
EVT_RETURN_LINK_KEYS	0x15	
EVT_PIN_CODE_REQ	0x16	
EVT_LINK_KEY_REQ	0x17	
EVT_LINK_KEY_NOTIFY	0x18	
EVT_LOOPBACK_COMMAND	0x19	
EVT_DATA_BUFFER_OVERFLOW	0x1A	
EVT_MAX_SLOTS_CHANGE	0x1B	
EVT_READ_CLOCK_OFFSET_COMPLETE	0x1C	
EVT_CONN_PTYPE_CHANGED	0x1D	
EVT_QOS_VIOLATION	0x1E	
EVT_PSCAN_REP_MODE_CHANGE	0x20	
EVT_FLOW_SPEC_COMPLETE	0x21	

EVT_INQUIRY_RESULT_WITH_RSSI	0x22	
EVT_READ_REMOTE_EXT_FEATURES_COMPLETE	0x23	
EVT_SYNC_CONN_COMPLETE	0x2C	
EVT_SYNC_CONN_CHANGED	0x2D	
EVT_SNIFF_SUBRATING	0x2E	
EVT_EXTENDED_INQUIRY_RESULT	0x2F	
EVT_ENCRYPTION_KEY_REFRESH_COMPLETE	0x30	
EVT_IO_CAPABILITY_REQUEST	0x31	
EVT_IO_CAPABILITY_RESPONSE	0x32	
EVT_USER_CONFIRM_REQUEST	0x33	
EVT_USER_PASSKEY_REQUEST	0x34	
EVT_REMOTE_OOB_DATA_REQUEST	0x35	
EVT_SIMPLE_PAIRING_COMPLETE	0x36	
EVT_LINK_SUPERVISION_TIMEOUT_CHANGED	0x38	
EVT_ENHANCED_FLUSH_COMPLETE	0x39	
EVT_USER_PASSKEY_NOTIFY	0x3B	
EVT_KEYPRESS_NOTIFY	0x3C	
EVT_REMOTE_HOST_FEATURES_NOTIFY	0x3D	

6.1.5 指令分组格式（0x01）

0	4	8	12	16	24	31
OpCode				Parameter Total Length	Parameter 0	
OCF			OGF			
Parameter 1				Parameter 2		
Parameter N-1				Parameter N		

OCF: Bit 0~9，取值见 OCF

OGF: Bit 10~15

6.1.6 事件分组格式（0x04）

0	4	8	12	16	24	31
Event Code			Parameter Total Length	Event Parameter 0		
Event Parameter 1				Event Parameter 2	Event Parameter 3	
Event Parameter N-1				Event Parameter N		

Event Code: 见 Event_Types

6.1.7 查找蓝牙设备

6.1.7.1 协议流程

Host -> BlueZ Inquiry Command

BlueZ ->Host Inquiry Result Event

BlueZ ->Host Inquiry Complete Event

6.1.7.2 Inquiry Command

Command	OCF	Command Parameters	Return Parameters
HCI_Inquiry	0x0001	LAP, Inquiry_Length, Num_Responses	

Command Parameters:

LAP:（Size:3 Bytes）

Value	Parameter Description
0x9E8B00–0X9E8B3F	This is the LAP from which the inquiry access code should be derived when the inquiry procedure is made;

Inquiry_Length: （Size:1 Bytes）

Value	Parameter Description

N = 0xXX	Maximum amount of time specified before the Inquiry is halted. Size: 1 octet Range: 0x01 – 0x30 Time = N * 1.28 sec Range: 1.28 – 61.44 Sec

Num_Responses: （Size:1 Bytes）

Value	Parameter Description
0x00	Unlimited number of responses
0xXX	Maximum number of responses from the Inquiry before the Inquiry is halted. Range: 0x01 – 0xFF

6.1.7.3 Inquiry Result Event

Event	Event Code	Event Parameters
Inquiry Result	0x02	Num_Responses, BD_ADDR[i], Page_Scan_Repetition_Mode[i], Reserved[i], Reserved[i], Class_of_Device[i] Clock_Offset[i]

Event Parameters:

Num_Responses: （Size:1 Bytes）

Value	Parameter Description
0xXX	Number of responses from the Inquiry

BD_ADDR[i]: （Size:6 Bytes * Num_Responses）

Value	Parameter Description
0xXXXXXXXXXXXX	BD_ADDR for each device which responded

Page_Scan_Repetition_Mode[i]: （Size:1 Bytes * Num_Responses）

Value	Parameter Description
0x00	R0
0x01	R1
0x02	R2
0x03-0xFF	Reserved

Reserved[i]: （Size:1 Bytes * Num_Responses）

Value	Parameter Description
0xXX	Reserved

Reserved[i]: （Size:1 Bytes * Num_Responses）

Value	Parameter Description
0xXX	Reserved , shall be set to 0x00.

Class_of_Device[i]: （Size:3 Bytes * Num_Responses）

Value	Parameter Description
0xXXXXXX	Class of Device for the device

Clock_Offset[i]: （Size:2 Bytes * Num_Responses）

Value	Parameter Description
Bit 14-0	Bit 16-2 of CLKslave-CLKmaster.
Bit 15	Reserved

6.1.7.4 Inquiry Complete Event

Event	Event Code	Event Parameters
Inquiry Complete	0x01	Status

Event Parameters:

Status: （Size:1 Bytes）

Value	Parameter Description
0x00	Inquiry command completed successfully
0x01-0xFF	Inquiry command failed.

6.1.8 获取蓝牙设备名称

6.1.8.1 协议流程

Host -> BlueZ Remote Name Request Command

BlueZ ->Host Remote Name Request Complete Event

6.1.8.2 Remote Name Request Command

Command	OCF	Command Parameters	Return Parameters
HCI_Remote_Name_Request	0x0019	BD_ADDR, Page_Scan_Repetition_Mode, Reserved, Clock_Offset	

Command Parameters:

BD_ADDR: （Size:6 Bytes）

Value	Parameter Description
0xXXXXXXXXXXXX	BD_ADDR for the device whose name is requested.

Page_Scan_Repetition_Mode: （Size:1 Bytes）

Value	Parameter Description
0x00	R0
0x01	R1
0x02	R2
0x03-0xFF	Reserved

Reserved: （Size:1 Bytes）

Value	Parameter Description
0x00	Reserved, must be set to 0x00

Clock_Offset: （Size:2 Bytes）

Value	Parameter Description
Bit 14.0	Bit 16.2 of CLKslave-CLKmaster.
Bit 15	Clock_Offset_Valid_Flag Invalid Clock Offset = 0 Valid Clock Offset = 1

6.1.8.3 Remote Name Request Complete Event

Event	Event Code	Event Parameters
Remote Name Request Complete	0x07	Status, BD_ADDR, Remote_Name

Event Parameters:

Status: （Size:1 Bytes）

Value	Parameter Description
0x00	Remote_Name_Request command succeeded.
0x01-0xFF	Remote_Name_Request command failed.

BD_ADDR: （Size:6 Bytes）

Value	Parameter Description
0xXXXXXXXXXXXX	BD_ADDR for the device whose name was requested.

Remote_Name: （Size:248 Bytes）

Value	Parameter Description
Name[248]	A UTF-8 encoded user-friendly descriptive name for the remote device. If the name contained in the parameter is shorter than 248 octets, the end of the name is indicated by a NULL octet (0x00), and the following octets (to fill up 248 octets, which is the length of the parameter) do not have valid values.

6.2 AT 指令

6.2.1 AT 指令格式

- AT 指令分隔符描述如下：

<cr>回车符

<lf>换行符

- 从 HF 到 AG 的指令格式为：

<AT command><cr>

- 从 AG 到 HF 的 OK 代码格式为：

<cr><lf>OK<cr><lf>

- 从 AG 到 HF 的 ERROR 代码格式为：

<cr><lf>ERROR<cr><lf>

- 从 AG 到 HF 的主动提供的结果代码的格式应为：

<cr><lf><result code><cr><lf>

6.2.2 AT+BRSF

- 描述：请求 AG 支持的功能，并传入 HF 支持的功能
- 语法：AT+BRSF=<HF 支持的指标值>
- 指标值 Bit 位：

Bit	Feature
0	EC and/or NR function
1	Call waiting and 3-way calling
2	CLI presentation capability
3	Voice recognition activation
4	Remote volume control
5	Enhanced call status
6	Enhanced call control
7	Codec negotiation
8	HF Indicators
9	eSCO S4(and T2) Settings Supported
10-31	Reserved for future definition

- 例子：

AT+BRSF=63

6.2.3 +BRSF

- 描述：返回 AG 支持的功能
- 语法：+BRSF: <AG 支持的指标值>
- 指标值 Bit 位：

Bit	Feature
0	Three-way calling
1	EC and/or NR function
2	Voice recognition activation
3	In-band ring tone capability
4	Attach a number to a voice tag
5	Ability to reject a call
6	Enhanced call status
7	Enhanced call control
8	Extended Error Result Codes
9-31	Reserved for future definition

- 例子：

+BRSF: 359

6.2.4 AT+CIND

- 描述：包括 AT+CIND?和 AT+CIND=?两个请求。

AT+CIND=? 请求返回 AG 的每个指标，以及它们的取值范围。

AT+CIND? 请求读取 AG 指标的当前状态值。

- 例子：

AT+CIND=?

AT+CIND?

6.2.5 +CIND

- 描述：返回 AT+CIND=?或 AT+CIND?两个请求的结果。
- 语法：

+CIND:(“<指标>”,(<指标最小值，指标最大值)),…)

+CIND:<指标当前状态值>,…

- 指标及值：

service 服务可用性指标

0 没有服务

1 有服务

call 标准呼叫状态指标

0 没有激活

1 有激活

callsetup 蓝牙专有的呼叫建立状态指标

0 当前不在呼叫设置中

1 正在进行呼入呼叫过程

2 正在进行呼出呼叫过程
3 远程调用中的远程方被警告
callheld 蓝牙专有的呼叫保持状态指标
0 没有调用
1 呼叫被搁置或激活持有呼叫交换
2 挂起，没有激活呼叫
signal 信号强度指标
范围是 0 到 5
roam 漫游状态指标
0 未激活
1 激活
battchg 电池充电指标
范围是 0 到 5

- 例子

AT+CIND=?的请求返回如下：
+CIND:
("call",(0,1)),("callsetup",(0-3)),("service",(0-1)),("signal",(0-5)),("roam",(0,1)),("battchg",(0-5)),("callheld",(0-2))
AT+CIND?的请求返回如下（返回的数值顺序同 AT+CIND=?请求）：
+CIND: 0,0,1,4,0,4,0

6.2.6 AT+CMER

- 描述：请求激活或解除事件报告

AT+CMER=3,0,0,1 激活指标事件报告
AT+CMER=3,0,0,0 解除指标事件报告

- 例子：

AT+CMER=3,0,0,1

6.2.7 AT+CLIP

- 描述：请求是否在线识别标准电话通知
- 语法：

AT+CLIP=<值>

- 值：

0 不启动
1 启动

- 例子：

AT+CLIP=1

6.2.8 AT+CCWA

描述：请求是否启动呼叫等待通知
语法：AT+CCWA=<值>
值：
0 不启动
1 启动
例子：
AT+CCWA =1

6.2.9 AT+CHLD

- 描述：包括 AT+CHLD=?和 AT+CHLD=<n>两种请求。

AT+CHLD=? 请求获取 AG 的标准呼叫保持和多方指挥处理
AT+CHLD=<n> 请求设置 AG 的标准呼叫保持和多方指挥处理，其中 n，可以是 0,1,1<idx>,2,2<idx>,3,4

- 例子：

AT+CHLD=?
AT+CHLD=1

6.2.10 +CHLD

- 描述：返回 AT+CHLD=?的结果
- 语法：

+CHLD:(<值>,…)

- 值：

0 释放所有持有电话或设置用户忙用户为等待电话
1 释放所有活动通话（如果有的话），并接受另一个（持有或等待）呼叫
1<idx> 释放指定的呼叫（<编号>）
2 将所有活动调用（如果存在）保留并接受另一个（持有）或等待）呼叫
2<idx> 请求私人咨询模式与指定的电话（<编号>）
3 为会话添加一个保留调用
4 连接两个电话和断开通话的用户（显式调用转移）

- 例子：

+CHLD: (0,1,2,3)

6.2.11 +CIEV

- 描述：返回指标当前的值
- 语法：+CIEV:<ind>,<value>

<ind> 指标索引，见 AT+CIND=?的请求返回的+CIND 的指标索引

<value> 指标索引的当前值

- 例子：

+CIEV: 4,3

6.2.12 ATD

描述：拨打电话指定号码

语法：

ATD<dd..dd>

<dd..dd> 电话号码

例子：

ATD19288813845

6.2.13 AT+BLDN

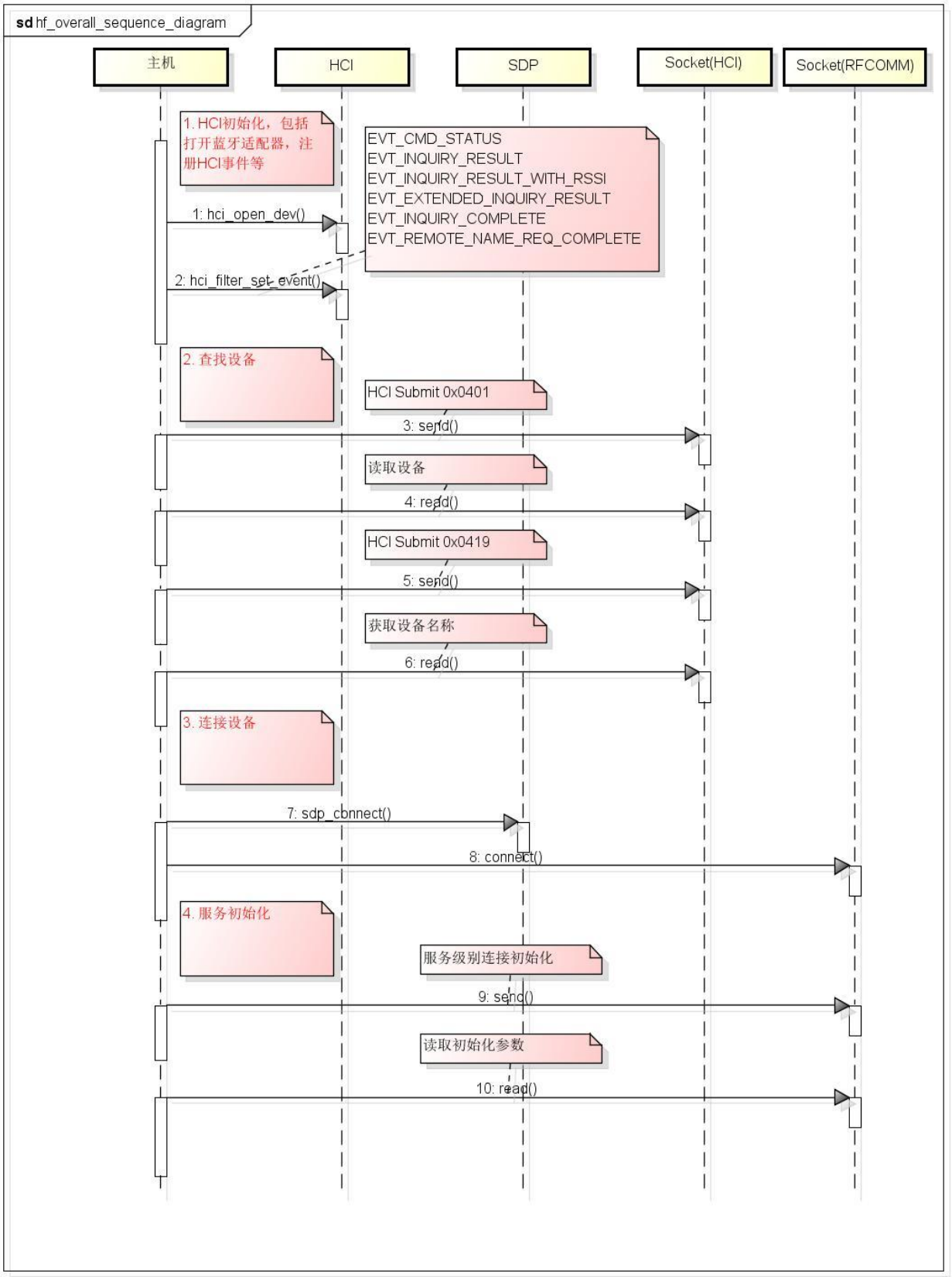
描述：拨打最后一个电话号码

语法：

AT+BLDN

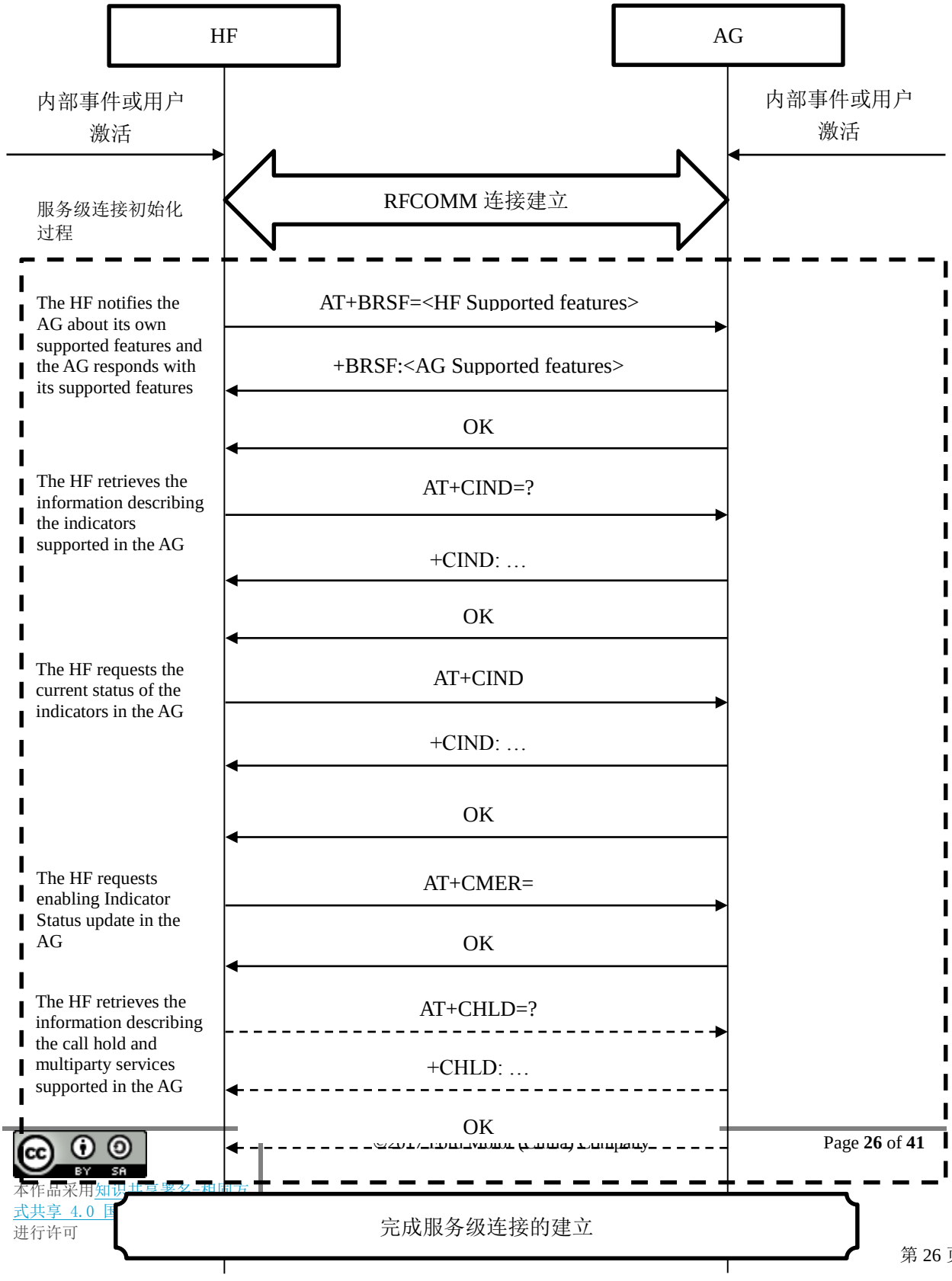
7 流程说明

7.1 主体流程

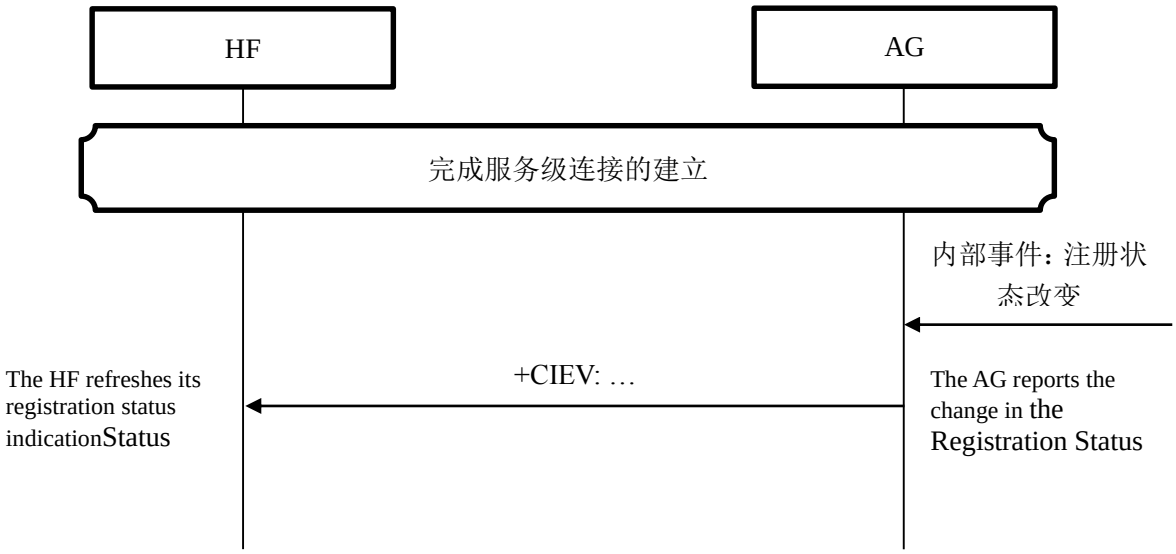


7.2 HFP 控制流程

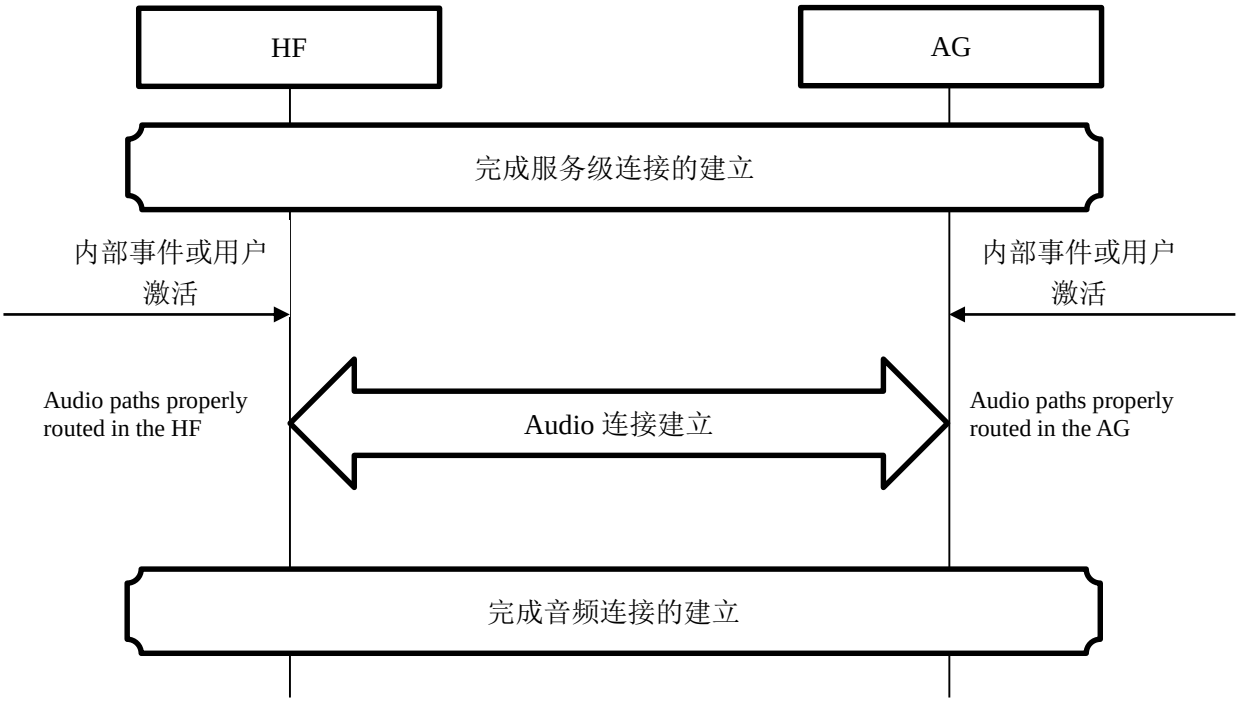
7.2.1 服务级连接建立



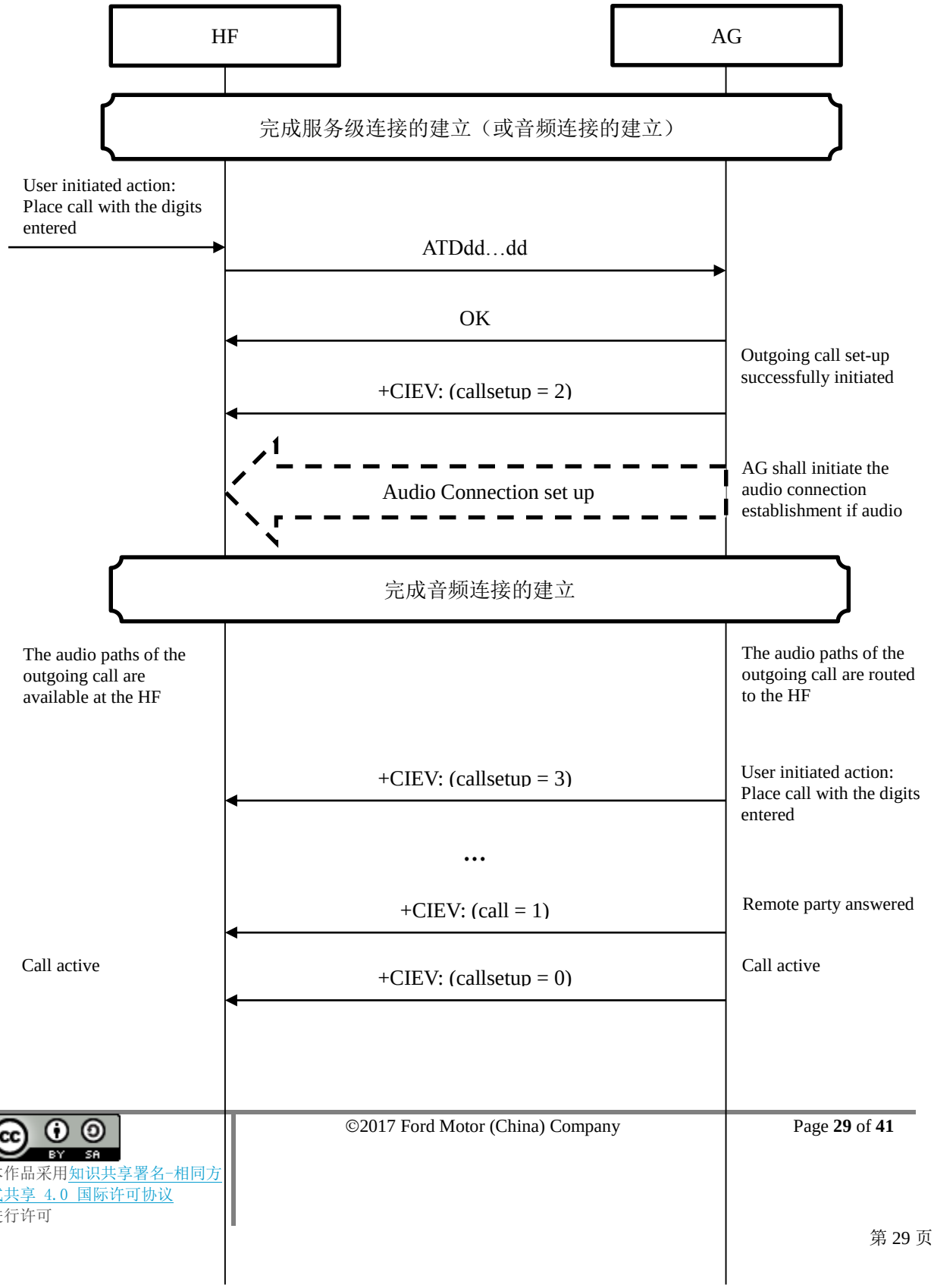
7.2.2 注册状态的传输



7.2.3 音频连接的建立



7.2.4 拨打电话



8 nohands 技术调研

8.1 概述

nohands 是一款基于 linux 系统的服务程序，它可以使 linux 系统成为一个移动电话的音频输入端或输出端。它的目标是兼容蓝牙 HFP1.5 所支持的所有指令和通知，以及音频流。

8.2 运行步骤

8.2.1 安装依赖库

sudo apt-get install subversion g++ autoconf libtool libspeexdsp-dev libasound2-dev libbluetooth-dev libaudiofile-dev libdbus-1-dev python-glade2

8.2.2 配置路径

在~/.zshrc 文件中添加
qt-x11 configure
export QT_DIR=/home/patrick/programs/qt/output/qt-x11
export PATH=$QT_DIR/bin:$PATH
export LD_LIBRARY_PATH=$QT_DIR/lib:$LD_LIBRARY_PATH
export PKG_CONFIG_PATH=/home/patrick/programs/qt/output/qt-x11/lib/pkgconfig
关闭保存.zshrc 文件。
source .zshrc

8.2.3 编译

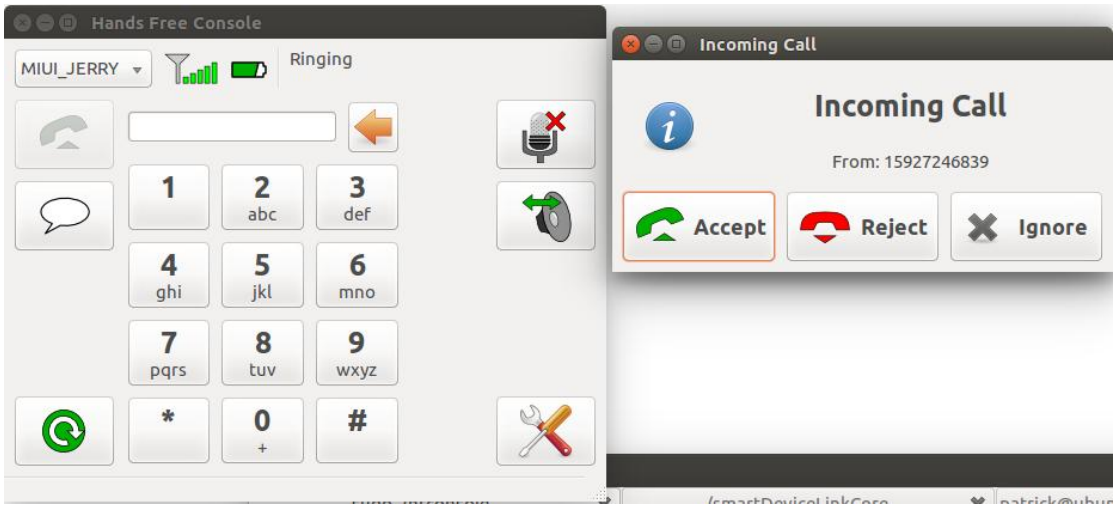
进入 nohands 目录。
./autogen.sh
./configure
make
sudo make install
（**********
如果 make 失败，尝试在 Makefile 文件中添加如下参数
*test/Makefile 文件修改
libhfp_LIBS = -lpthread -lbluetooth -lasound -laudiofile -lspeexdsp
CXXFLAGS = -g -Wall -fpermissive
*libhfp/Makefile 文件修改

libhfp_LIBS = -lpthread -lbluetooth -lasound -laudiofile -lspeexdsp
CXXFLAGS = -g -Wall -fpermissive
*hfpd/Makefile 文件修改
libhfp_LIBS = -lpthread -lbluetooth -lasound -laudiofile -lspeexdsp
CXXFLAGS = -g -Wall -fpermissive
**********)

8.2.4 运行

进入 nohands/data 目录，运行 hfconsole 程序，如下：
cd data
sudo ./hfconsole
界面如下：

主界面



拨打/接听电话界面

8.2.5 问题及解决

8.2.5.1 如何让修改的代码生效

如果修改了 libhfp 或者 hfpd 目录的代码，则需要做如下操作，才能生效：

*杀掉守护进程/usr/local/bin/hfpd

ps -aux

kill -9 [进程 id]

*重新编译

make

sudo make install

8.2.5.2 解决 sdp_connect(BDADDR_ANY，BDADDR_LOCAL，SDP_RETRY_IF_BUSY)返回 NULL 的问题

*添加 compat 选项

sudo vi /lib/systemd/system/bluetooth.service

修改 ExecStart=/usr/libexec/bluetooth/bluetoothd --compat

sudo systemctl daemon-reload

sudo service bluetooth restart

设置 sdp 文件的访问权限

sudo chmod 777 /var/run/sdp

设置 root 权限运行

sudo ./hfconsole

8.2.5.3 解决无法搜索到蓝牙手机设备的问题

修改的 diff 文件内容如下：

From d55087336a1a7a7e3f314b0a9942d09a4dec10cd Mon Sep 17 00:00:00 2001

From: ilikegithubcom <panliang01@beyondsoft.com>

Date: Fri, 20 Oct 2017 17:52:03 +0800

Subject: [PATCH] add extended_inquiry_result handler

---

libhfp/bt.cpp | 127 ++---------

1 file changed, 108 insertions(+), 19 deletions(-)

diff --git a/libhfp/bt.cpp b/libhfp/bt.cpp

index bae4836..1876b90 100644

--- a/libhfp/bt.cpp

+++ b/libhfp/bt.cpp

@@ -68,6 +68,13 @@ namespace libhfp {

#define SDP_ATTR_SUPPORTED_FEATURES SDP_SUPPORTED_FEATURES

```
 #endif

+// inquiry result type
+typedef enum{
+    inq_result_default = 0,
+    inq_result_rssi,
+    inq_result_externed
+}e_inq_result_type;
+
 int SdpAsyncTaskHandler::
 SdpLookupChannel(SdpTaskParams &htp)
 {
@@ -426,15 +433,20 @@ HciDataReadyNot(SocketNotifier *notp, int fh)
       HciTask *taskp;
       inquiry_info *infop = 0;
       inquiry_info_with_rssi *rssip = 0;
+     extended_inquiry_info *exinfop = 0;
       uint8_t count = 0;
       ssize_t ret;
-     bool inq_result_rssi = false;
+     e_inq_result_type inq_result_type = inq_result_default;
       ErrorInfo error;

       assert(fh == m_hci_fh);
       assert(notp == m_hci_not);

       ret = read(m_hci_fh, evbuf, sizeof(evbuf));
+//    GetDi()->LogDebug("evbuf: ");
+//    for(int i = 0; i < sizeof(evbuf); i++){
+//        GetDi()->LogDebug("0x%02x ", evbuf[i]);
+//    }
       if (ret < 0) {
             if ((errno == EAGAIN) ||
                   (errno == EINTR) ||
@@ -460,8 +472,10 @@ HciDataReadyNot(SocketNotifier *notp, int fh)
             return;
       }
```

```
-       if (evbuf[0] != HCI_EVENT_PKT)
+       if (evbuf[0] != HCI_EVENT_PKT){
+//          GetDi()->LogDebug("if ((errno == EAGAIN) || evbuf[0]: %d", evbuf[0]);
                return;
+       }

        if (ret < (1 + HCI_EVENT_HDR_SIZE)) {
                GetDi()->LogError(&error,
@@ -550,24 +564,30 @@ HciDataReadyNot(SocketNotifier *notp, int fh)
                countp = (uint8_t *) (hdr + 1);
                infop = (inquiry_info *) (countp + 1);
                ret = 1;
-               if (hdr->plen < ret)
+               if (hdr->plen < ret){
+//                  GetDi()->LogDebug("if (hdr->plen < ret){");
                        goto invalid_struct;
+               }
                count = *countp;
                ret = 1 + (count * sizeof(*infop));
-               if (hdr->plen != ret)
+               if (hdr->plen != ret){
+//                  GetDi()->LogDebug("if (hdr->plen != ret){");
                        goto invalid_struct;
+               }

-               inq_result_rssi = false;
-
-       do_next_inq:
+               inq_result_type = inq_result_default;
+// GetDi()->LogDebug("case EVT_INQUIRY_RESULT: %d", count);
+       do_next_inq_default:
                if (!count)
                        break;
                listp = m_hci_tasks.next;
                while (listp != &m_hci_tasks) {
+//                  GetDi()->LogDebug("while (listp != &m_hci_tasks) {");
                        taskp = GetContainer(listp, HciTask, m_hcit_links);
                        listp = listp->next;
```

```
                if (taskp->m_tasktype == HciTask::HT_INQUIRY) {
+//              GetDi()->LogDebug("if (taskp->m_tasktype == HciTask::HT_INQUIRY) {");
                        taskp->m_complete = false;
                        bacpy(&taskp->m_bdaddr, &infop->bdaddr);
                        taskp->m_pscan = infop->pscan_mode;
@@ -588,15 +608,19 @@ HciDataReadyNot(SocketNotifier *notp, int fh)
            countp = (uint8_t *) (hdr + 1);
            rssip = (inquiry_info_with_rssi *) (countp + 1);
            ret = 1;
-           if (hdr->plen < ret)
+           if (hdr->plen < ret){
+//              GetDi()->LogDebug("if (hdr->plen < ret){");
                goto invalid_struct;
+           }
            count = *countp;
            ret = 1 + (count * sizeof(*rssip));
-           if (hdr->plen != ret)
+           if (hdr->plen != ret){
+//              GetDi()->LogDebug("if (hdr->plen != ret){");
                goto invalid_struct;
+           }

-           inq_result_rssi = true;
-
+           inq_result_type = inq_result_rssi;
+// GetDi()->LogDebug("case EVT_INQUIRY_RESULT_WITH_RSSI: %d", count);
      do_next_inq_rssi:
            if (!count)
                break;
@@ -620,6 +644,50 @@ HciDataReadyNot(SocketNotifier *notp, int fh)
            }
            break;
      }
+     case EVT_EXTENDED_INQUIRY_RESULT: {
+           uint8_t *countp;
+           countp = (uint8_t *) (hdr + 1);
+           exinfop = (extended_inquiry_info *) (countp + 1);
```

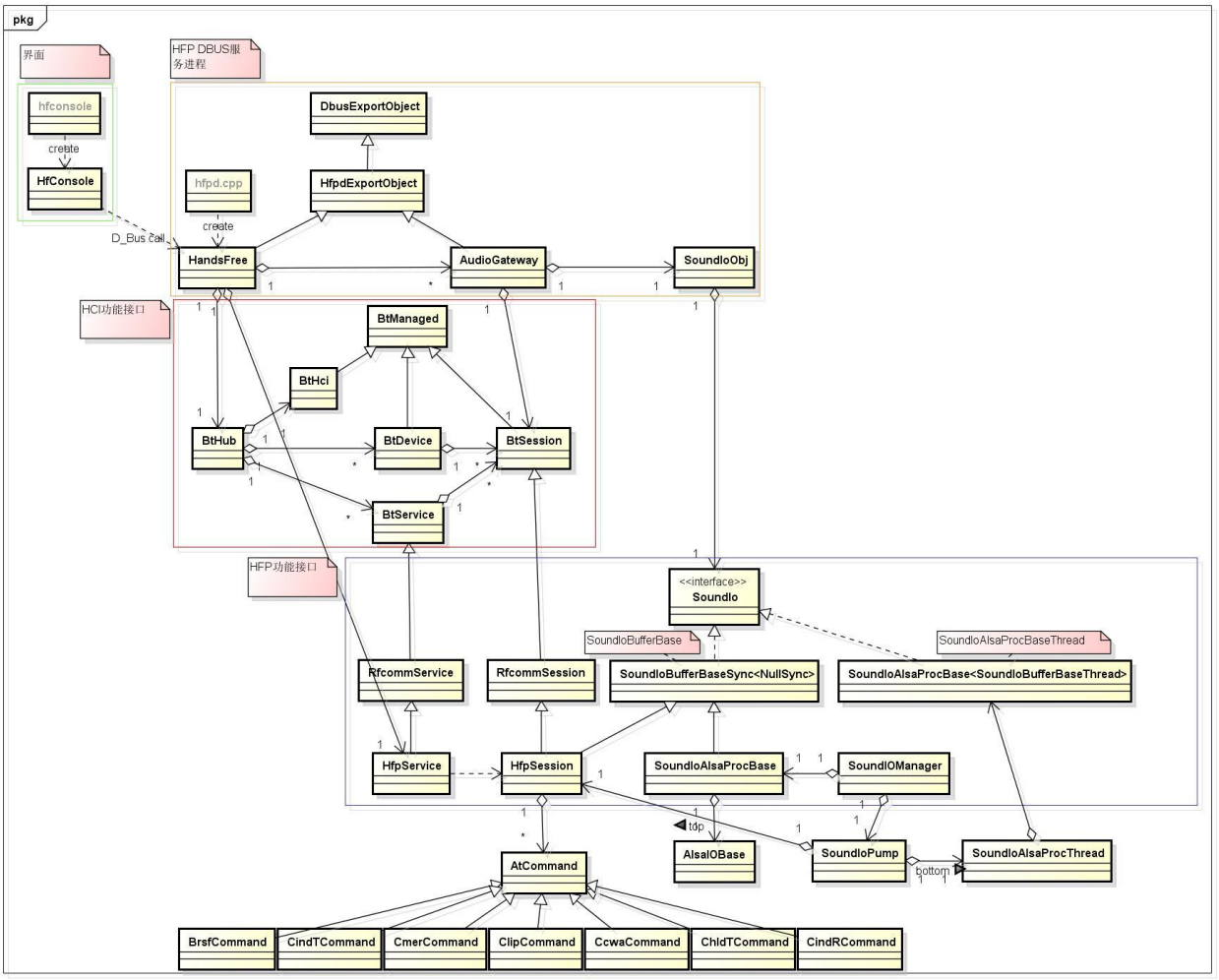
```
+            ret = 1;
+            if (hdr->plen < ret){
+//              GetDi()->LogDebug("if (hdr->plen < ret){");
+                  goto invalid_struct;
+            }
+            count = *countp;
+            ret = 1 + (count * sizeof(*exinfop));
+            if (hdr->plen != ret){
+//              GetDi()->LogDebug("if (hdr->plen != ret){");
+                  goto invalid_struct;
+            }
+
+            inq_result_type = inq_result_externed;
+// GetDi()->LogDebug("case EVT_EXTENDED_INQUIRY_RESULT: %d", count);
+     do_next_inq_extended:
+            if (!count)
+                  break;
+            listp = m_hci_tasks.next;
+            while (listp != &m_hci_tasks) {
+//              GetDi()->LogDebug("while (listp != &m_hci_tasks) {");
+                  taskp = GetContainer(listp, HciTask, m_hcit_links);
+                  listp = listp->next;
+
+                  if (taskp->m_tasktype == HciTask::HT_INQUIRY) {
+//              GetDi()->LogDebug("if (taskp->m_tasktype == HciTask::HT_INQUIRY) {");
+                        taskp->m_complete = false;
+                        bacpy(&taskp->m_bdaddr, &exinfop->bdaddr);
+                        taskp->m_pscan = 0;//exinfop->pscan_mode;
+                        taskp->m_pscan_rep = exinfop->pscan_rep_mode;
+                        taskp->m_clkoff = exinfop->clock_offset;
+                        taskp->m_devclass =
+                              (exinfop->dev_class[2] << 16) |
+                              (exinfop->dev_class[1] << 8) |
+                              exinfop->dev_class[0];
+                        taskp->m_hcit_links.UnlinkOnly();
+                        tasks_done.AppendItem(taskp->m_hcit_links);
+                  }
+            }
```

```
+         break;
+     }
      case EVT_INQUIRY_COMPLETE: {
            uint8_t st;
            ret = 1;
@@ -688,12 +756,28 @@ HciDataReadyNot(SocketNotifier *notp, int fh)
      }

      if (count--) {
-           if (inq_result_rssi) {
-                 rssip++;
-                 goto do_next_inq_rssi;
-           }
-           infop++;
-           goto do_next_inq;
+       switch(inq_result_type){
+           case inq_result_default:
+           {
+              infop++;
+              goto do_next_inq_default;
+              break;
+           }
+           case inq_result_rssi:
+           {
+              rssip++;
+              goto do_next_inq_rssi;
+              break;
+           }
+           default:
+           {
+              exinfop++;
+              goto do_next_inq_extended;
+              break;
+           }
+
+       }
+
      }
```

```
      return;
@@ -707,7 +791,6 @@ invalid_struct:
      GetHub()->InvoluntaryStop(&error);
  }

-
  bool BtHci::
  HciSend(int fh, HciTask *taskp, void *data, size_t len, ErrorInfo *error)
  {
@@ -735,7 +818,12 @@ HciSend(int fh, HciTask *taskp, void *data, size_t len, ErrorInfo *error)
      GetDi()->LogDebug("HCI Submit 0x%04x", hdrp->opcode);

      while (1) {
-         ret = send(fh, buf, expect, MSG_NOSIGNAL);
+       //>>>>>send>>>>>//
+//       for(int i = 0; i < expect; i++){
+//          GetDi()->LogDebug("0x%x ",buf[i]);
+//       }
+       ret = send(fh, buf, expect, MSG_NOSIGNAL);
+       //<<<<<send<<<<<//
          if ((ret < 0) &&
              ((errno == EAGAIN) ||
               (errno == EINTR) ||
@@ -859,6 +947,7 @@ HciInit(int hci_id, ErrorInfo *error)
      hci_filter_set_event(EVT_CMD_STATUS, &flt);
      hci_filter_set_event(EVT_INQUIRY_RESULT, &flt);
      hci_filter_set_event(EVT_INQUIRY_RESULT_WITH_RSSI, &flt);
+     hci_filter_set_event(EVT_EXTENDED_INQUIRY_RESULT, &flt);
      hci_filter_set_event(EVT_INQUIRY_COMPLETE, &flt);
      hci_filter_set_event(EVT_REMOTE_NAME_REQ_COMPLETE, &flt);

--
```

8.3 逻辑结构

8.3.1 总体结构



8.4 参考网站

https://github.com/heinervdm/nohands
http://nohands.sourceforge.net/