# 兰桂SDK开发说明-蓝牙小车控制

**SDK主要功能：**

响应第三方设备请求，管理蓝牙路由，扩展连接范围，本地或云端最多支持22个设备，SDK可以直接集成到本地应用/服务器中，基于HTPP/HTPPS实现，支持c#/Node.js/java/python等语言，读/写蓝牙设备数据，创建远程调度。

**通信流程如下：**

Cassia AP的工作流程可以简单描述为：Cassia AP使用HTTP协议，向指定的蓝牙端口写入信息。调试HTTP协议时可以使用Fiddler软件。



**使用HTTP方法，具体方法如下：**

|  |  |
| --- | --- |
| 项目 | 获取设备HOST IP |
| HTTP方法 | GET |
| URL | http://api.cassianetworks.com/cassia/hubs/CC:1B:E0:E0:25:90 |
| 返回示例 | **{**  "mac"**:**"CC:1B:E0:E0:25:90"**,**  "ip"**:**"172.16.3.110"**,**  "chipId"**:[**  "0"**,**  "1"  **],**  "name"**:**"CassiaHub-undefined"**,**  "version"**:**"ToB\_1.0.2\_201703011548"**,**  "sdkV"**:**2.2  **}** |
| 项目 | 获取设备状态 |
| HTTP方法 | GET |
| URL | http://<HOST-IP>/cassia/info/?mac=CC:1B:E0:E0:25:90 |
| 返回示例 | **{**  "ip"**:**"172.16.3.110"**,**  "mac"**:**"CC:1B:E0:E0:25:90"**,**  "wired"**:{**  "proto"**:**"dhcp"  **},**  "wireless"**:{**  "ssid"**:**"AnChuangKongJian"**,**  "password"**:**"jiasuqi@0111"  **},**  "networkType"**:**"wireless"**,**  "timeZone"**:**"Asia/Shanghai"**,**  "versionInfo"**:{**  "Hard ver"**:**"1.0"**,**  "group"**:**" "**,**  "Running Version"**:**"A"**,**  "Count\_A"**:**"2"**,**  "Count\_B"**:**"1"**,**  "Kernel\_Ver\_A"**:**"ToB\_1.0.1\_201606231552"**,**  "Sys\_Ver\_A"**:**"ToB\_1.0.2\_201703011548"**,**  "Kernel\_Ver\_B"**:**"1.0.1\_201701111740"**,**  "Sys\_Ver\_B"**:**"1.0.1\_201701111740"**,**  "System uptime"**:**"1 hours"  **}**  **}** |
| 项目 | 连接BLE设备 |
| HTTP方法 | POST |
| URL | http://<HOST-IP>/gap/nodes/<BLE-MAC>/connection?mac=CC:1B:E0:E0:25:90&chip=1 |
| Header | Content-Type: application/json  version: 1  Host: 172.16.1.12  Content-Length: 49 |
| Body | **{**  "timeOut"**:** "100"**,**  "type" **:**"random"  **}** |
| 返回示例 | OK |
| 项目 | 获取连接设备 |
| HTTP方法 | GET |
| URL | http://<HOST-IP>/gap/nodes/?connection\_state=connected&mac=CC:1B:E0:E0:25:90 |
| 返回示例 | **{**  "nodes"**:[**  **{**  "handle"**:**""**,**  "bdaddrs"**:{**  "bdaddr"**:**""**,**  "bdaddrType"**:**""  **},**  "connectionState"**:**"connected"**,**  "name"**:**""**,**  "id"**:**"00:0E:0B:53:4D:48"**,**  "type"**:**""**,**  "chipId"**:**"1"  **}**  **]**  **}** |
| 项目 | 断开BLE连接 |
| HTTP方法 | DELETE |
| URL | http://<HOST-IP>/gap/nodes/<BLE-MAC>/connection?mac=CC:1B:E0:E0:25:90 |
| 返回示例 | OK |
| 项目 | 发现服务 |
| HTTP方法 | GET |
| URL | http://<HOST-IP>/gatt/nodes/<BLE-MAC>/services/?mac=CC:1B:E0:E0:25:90&all=1&uuid= |
| 返回示例 | **{**  "services"**:[**  **{**  "handle"**:**1**,**  "uuid"**:**"00001800-0000-1000-8000-00805f9b34fb"**,**  "primary"**:true**  **},**  **{**  "handle"**:**12**,**  "uuid"**:**"00001801-0000-1000-8000-00805f9b34fb"**,**  "primary"**:true**  **},**  **{**  "handle"**:**16**,**  "uuid"**:**"0000ffe0-0000-1000-8000-00805f9b34fb"**,**  "primary"**:true**  **}**  **]**  **}** |
| 项目 | 按线程推送数据 |
| HTTP方法 | GET |
| URL | http://<HOST-IP>/gatt/nodes/<BLE-MAC>/handle/<HANDLE>/value/<VALUE>/?mac=CC:1B:E0:E0:25:90 |
| 返回示例 | OK |

**注释：**

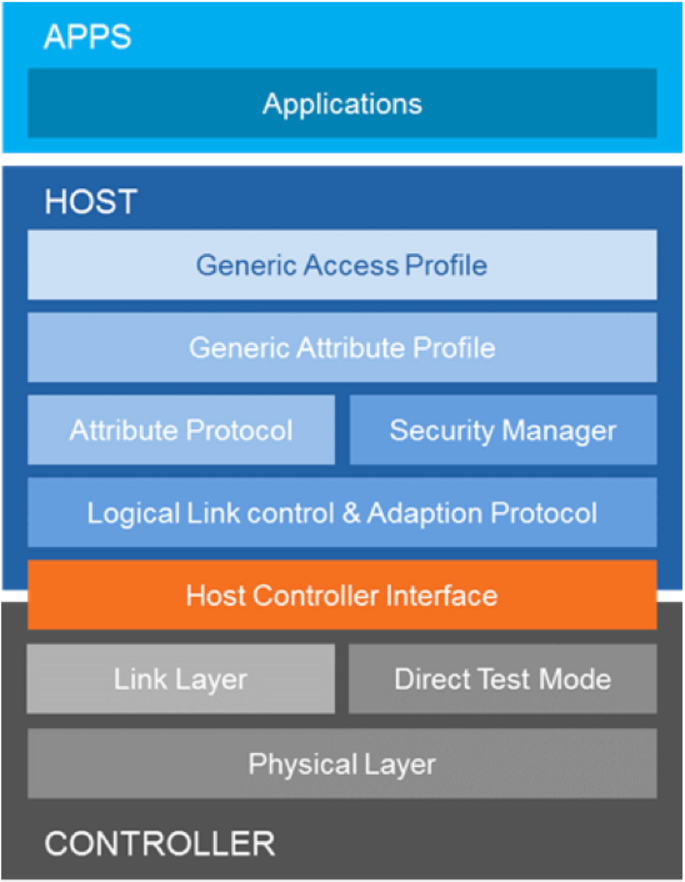
<HOST-IP> :获取的Cassia AP网络服务器IP地址。

<BLE-MAC> :要连接的蓝牙BLE设备MAC地址。

<HANLE> :Cassia AP根据Service UUID所建立的索引号，再查看BLE设备的Service UUID时，所使用的串口蓝牙模块的UUID为：0000ffe0-0000-1000-8000-00805f9b34fb。

<VALUE> :向service写入的数据。

**BLE协议栈**



从下到上分别为：控制层（CONTROLLER）→主机层（HOST）→应用层（APPS）。

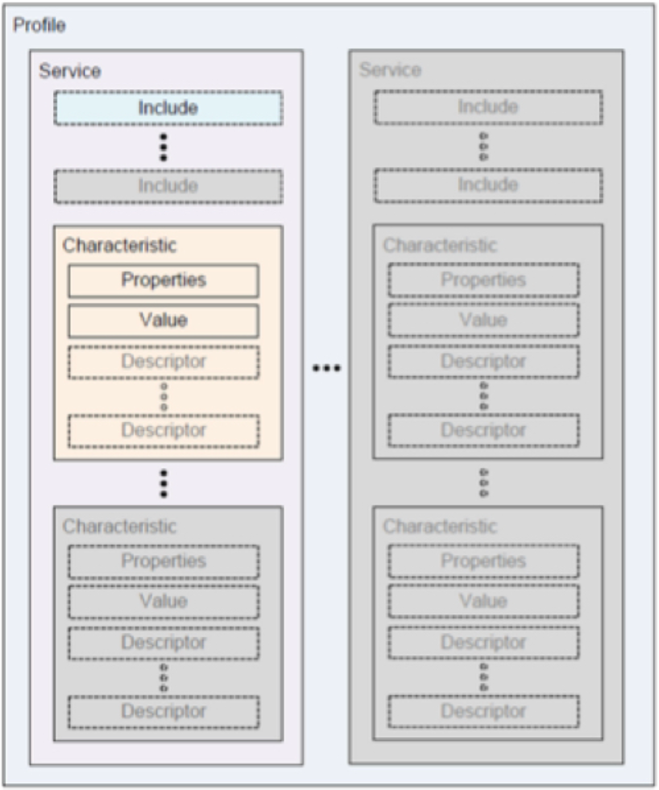
控制层：协议栈的底层实现，直接与硬件相关，一般直接集成在 SoC 中，由芯片厂商实现，包括物理层和链路层。

主机层：协议栈的上层实现，是硬件的抽象，与具体的硬件和厂家无关。

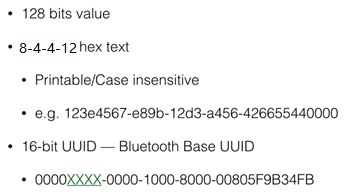
应用层：使用主机层提供的API，开发的应用。

**GATT**

GATT，全称叫做通用属性配置，它是建立ATT（Attribute Protocol）的基础上，对ATT进行进一步的逻辑封装，定义数据的交互方式和含义。GATT按照层级定义了三个概念：服务（Service）、特征（Characteristic）和描述（Descriptor），关系如下图所示：一个Service包含若干个 Characteristic，一个Characteristic可以包含若干Descriptor。而Characteristic定义了数值和操作。Characteristic的操作这几种权限：读、写、通知等权限。我们说的BLE通信，其实就是对Characteristic的读写或者订阅通知。还有最外面一层，Profile配置文件，把若干个相关的Service组合在一起，就成为了一个Profile，这样Profile就定义了一个实际的应用场景。



Service、Characteristic和Descriptor都是使用UUID唯一标示。UUID是全局唯一标识，它是128bit的值，为了便于识别和阅读，一般标示为8-4-4-4-12的如下16进制形式。



为避免冲突，一般都不会自己手动定义UUID，Android中提供UUID.randomUUID()来生成一个随机的UUID。UID有点长，在低功耗蓝牙中数据长度非常受限的情况下，蓝牙又使用了16bit或者32bit的UUID。蓝牙SIG协会定义了一个基础的UUID（Bluetooth Base UUID），除了XXXX几位外，其他都是固定，所以16bit UUID应了一个128bit的UUID。但随之而来的问题就是，重可定义UUID数大幅减少，例如16bit UUID只有65536个，所以16bit UUID 并不能随便定义。

**HM-13蓝牙双模模块**

使用的HM-13蓝牙双模模块的service UUID：



利用发现服务可以看到如下信息：

**{**

"handle"**:**16**,**

"uuid"**:**"0000ffe0-0000-1000-8000-00805f9b34fb"**,**

"primary"**:true**

**}**

HM-13 BLE的服务UUID对应的handle号为16，后面利用此handle号发送服务信息。

**蓝牙小车**

小车控制指令如下：

|  |  |
| --- | --- |
| 前进 | $1,0,0,0,0,0,0,0,0,0#$0,0,0,0,0,0,0,0,0,0# |
| 后退 | $2,0,0,0,0,0,0,0,0,0#$0,0,0,0,0,0,0,0,0,0# |
| 左转 | $3,0,0,0,0,0,0,0,0,0#$0,0,0,0,0,0,0,0,0,0# |
| 右转 | $4,0,0,0,0,0,0,0,0,0#$0,0,0,0,0,0,0,0,0,0# |
| 停止 | $0,0,0,0,0,0,0,0,0,0# |
| 停止上传 | $0,0,0,2,0,0,0,0,0,0# |

利用按线程推送服务：

http://<HOST-IP>/gatt/nodes/<BLE-MAC>/handle/<HANDLE>/value/<VALUE>/?mac=

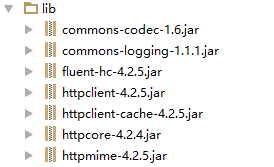
将<VALUE>替换为实际的控制指令即可。由于http协议不支持特殊字符，故需要将以上指令转换为ASCII码表上对应的HEX值。

小车控制指令ASCII -> HEX：

|  |  |
| --- | --- |
| 前进 | 24312c302c302c302c302c302c302c302c302c302324302c302c302c302c302c302c302c302c302c3023 |
| 后退 | 24322c302c302c302c302c302c302c302c302c302324302c302c302c302c302c302c302c302c302c3023 |
| 左转 | 24332c302c302c302c302c302c302c302c302c302324302c302c302c302c302c302c302c302c302c3023 |
| 右转 | 24342c302c302c302c302c302c302c302c302c302324302c302c302c302c302c302c302c302c302c3023 |
| 停止 | 24302c302c302c302c302c302c302c302c302c3023 |

**Android APP**

安卓APP端通过调用apache http服务包实现http方法。



下面为HTTP GET实现的方法

//HTTP GET方法

public static String sendHttpGet**(**String url**)** **{**

String result **=** **null;**

HttpClient httpClient **=** **new** DefaultHttpClient**();**

HttpGet httpGet **=** **new** HttpGet**(**url**);**

**try** **{**

HttpResponse httpResponse **=** httpClient**.**execute**(**httpGet**);**

**if** **(**httpResponse**.**getStatusLine**().**getStatusCode**()** **==** 200**)** **{**

HttpEntity entity **=** httpResponse**.**getEntity**();**

String response **=** EntityUtils**.**toString**(**entity**,** "utf-8"**);** //将entity当中的数据转换为字符串

result **=** response**.**toString**();**

**}**

**}** **catch** **(**Exception e**)** **{**

e**.**printStackTrace**();**

**}**

**return** result**;**

**}**

下面为HTTP POST实现的方法

//HTTP POST方法

public static String sendHttpPost**(**String url**)** **{**

String result **=** **null;**

**try** **{**

JSONObject jsonObj **=** **new** JSONObject**();**

jsonObj**.**put**(**"timeOut"**,** "300"**);**

jsonObj**.**put**(**"type"**,** "random"**);**

HttpClient httpClient **=** **new** DefaultHttpClient**();**

HttpPost httpPost **=** **new** HttpPost**(**url**);**

httpPost**.**setHeader**(**"Content-Type"**,** "application/json"**);**

HttpEntity requestEntity **=** **new** StringEntity**(**jsonObj**.**toString**());**

httpPost**.**setEntity**(**requestEntity**);**

**try** **{**

HttpResponse response **=** httpClient**.**execute**(**httpPost**);**

**if** **(**response**.**getStatusLine**().**getStatusCode**()** **==** 200**)** **{**

HttpEntity entity **=** response**.**getEntity**();**

BufferedReader reader **=** **new** BufferedReader**(**

**new** InputStreamReader**(**entity**.**getContent**()));**

result **=** reader**.**readLine**();**

**}**

**}** **catch** **(**Exception e**)** **{**

**}**

**}** **catch** **(**Exception e**)** **{**

**}**

**return** result**;**

**}**

软件git库：git@172.16.1.229:/var/git/bleapcart.git