veepooSDK 接入文档

密级程度：仅供合作方使用

编辑：李金亮

时间：2017-03-23

接入流程说明

因为 Android 只要在系统 4.3 以上才支持 BLE 4.0,所以创建项目时选择 API 最低版本要求是 19.

想最快了解 SDK 的调用,您可运行提供的 VpBluetoothSDKDemo 项目

具体接入的操作顺序是 导入 Jar 包--->配置 AndroidMainfexst.xml->调用 SDK。

具体调用 SDK 的方法说明,请查看 ApiDoc 文件下的 index.html.

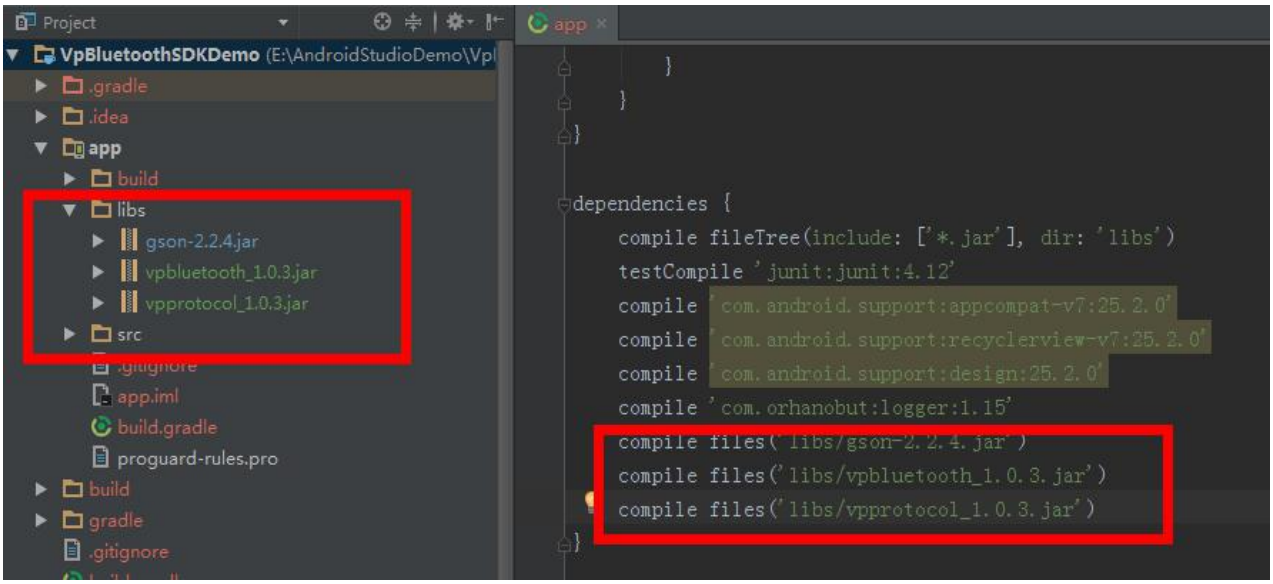
查看 api 时，您只从 VPOperateManager 类开始查看即可。
VPOperateManager 在 com.veepoo.protocol 下

一、引用 lib

在 app 中引用以下三个 lib

1.gson-2.2.4.jar
2.vpbluetooth_x.x.x.jar
3.vpprotocol_x.x.x.jar

如图：



引用成功进行下一步操作

二、配置 Androidmanifest.xml

1.权限配置

包含蓝牙、位置信息、网络、读写外部存储文件权限。

在一些 Andorid6.0 的手机上只有打开了位置信息权限，才到扫描到设备。

```
<uses-permission android:name="android.permission.BLUETOOTH" />
<uses-permission android:name="android.permission.BLUETOOTH_ADMIN" />
<uses-permission android:name="android.permission.ACCESS_FINE_LOCATION" />
<uses-permission android:name="android.permission.ACCESS_COARSE_LOCATION" />
<uses-permission android:name="android.permission.INTERNET" />
<uses-permission android:name="android.permission.ACCESS_NETWORK_STATE" />
<uses-permission android:name="android.permission.WRITE_EXTERNAL_STORAGE" />
```

2.软硬件特性要求

硬件要支持蓝牙。

```
<uses-feature
        android:name="android.hardware.bluetooth_le"
        android:required="true" />
```

3.Activity&Service 配置

```
<!--蓝牙服务-->
<service android:name="com.inuker.bluetooth.library.BluetoothService" />
<!--固件升级功能相关,具体可以查看 VpBluetoothSDKDem 源码-->
<service android:name=".oad.service.DfuService" />
<activity android:name=".oad.activity.NotificationActivity" />
```

三、进行蓝牙通信连接

1.操作说明

所有的操作都只是通过 VPOperateManager；

获取 VPOperateManager 实例

注意：为了避免内存泄漏的情况，请谨慎使用 context,推荐使用 getApplicationContext();

Context context=getApplicationContext()
VPOperateManager.getMangerInstance(context)；

具体调用 SDK 的方法说明,请查看 ApiDoc 文件下的 index.html.

比如调用扫描设备

VPOperateManager.getMangerInstance().startScanDevice()；

以下统一简写为

startScanDevice();

配置 Lib 成功后，需要顺序进行以下操作，具体的方法说明请参考 ApiDoc.

2.扫描蓝牙设备

startScanDevice();

3.连接蓝牙设备

connectDevice();

4.设置连接设备的连接状态监听

registerConnectStatusListener(); 这个方法最好是在连接成功后设置

5.设置系统蓝牙的开关状态监听

registerBluetoothStateListener(); 这个方法可以在任何状态下设置

四、进行蓝牙数据交互

蓝牙通信连接成功后

第一步要进行的交互是验证密码：confirmDevicePwd()
第二步要进行的交互是同步个人信息：syncPersonInfo()

其他具体的数据交互案例可参考我司的运行 demo 工程 VpBluetoothSDKDemo 下的 com.timaimee.vpbluetoothsdkdemo.activity.OperaterActivity

1.蓝牙日志说明

因为 SDK 的设计分为两层[蓝牙层、自定义协议层],为了区别日志,方便分析,这两层分别用了不一样的 Tag.
Tag 为 veepoo-bluetooth 表示蓝牙层日志;veepoo-profile:表示自定义协议层日志;
关闭日志：VPLogger.setDebug(false)

2.蓝牙数据下发

SDK 在蓝牙数据的下发的设计是只需要调用方法,传入参数,传入监听

以 confirmDevicePwd 为例

```
String   pwdStr=“0000”;
boolean is24Hourmodel = false;
VPOperateManager.getMangerInstance(mContext).confirmDevicePwd(writeResponse, new IPwdDataListener() {
                    @Override
                    public void onPwdDataChange(PwdData pwdData) {
                        String message = "PwdData:\n" + pwdData.toString();
                        Logger.t(TAG).i(message);
                    }
                }, new IDeviceFuctionDataListener() {
                    @Override
                    public void onFunctionSupportDataChange(FunctionDeviceSupportData functionSupport) {
                        String message = "FunctionDeviceSupportData:\n" + functionSupport.toString();
                        Logger.t(TAG).i(message);
                    }
                }, new ISocialMsgDataListener() {
                    @Override
                    public void onSocialMsgSupportDataChange(FunctionSocailMsgData socailMsgData) {
                        String message = "FunctionSocailMsgData:\n" + socailMsgData.toString();
                        Logger.t(TAG).i(message);
                    }
                }, pwdStr, is24Hourmodel);
```

3.蓝牙数据返回

SDK 在蓝牙数据的返回的设计是传入数据监听，数据有返回时会触发回调。

以 confirmDevicePwd 为例

```
String    pwdStr=“0000”;
boolean is24Hourmodel = false;
VPOperateManager.getMangerInstance(mContext).confirmDevicePwd(writeResponse, new IPwdDataListener() {
                    @Override
                    public void onPwdDataChange(PwdData pwdData) {
                        String message = "PwdData:\n" + pwdData.toString();
                        Logger.t(TAG).i(message);
                    }
                }, new IDeviceFuctionDataListener() {
                    @Override
                    public void onFunctionSupportDataChange(FunctionDeviceSupportData functionSupport) {
                        String message = "FunctionDeviceSupportData:\n" + functionSupport.toString();
                        Logger.t(TAG).i(message);
                    }
                }, new ISocialMsgDataListener() {
                    @Override
                    public void onSocialMsgSupportDataChange(FunctionSocailMsgData socailMsgData) {
                        String message = "FunctionSocailMsgData:\n" + socailMsgData.toString();
                        Logger.t(TAG).i(message);
                     }
                }, pwdStr, is24Hourmodel);
```

五、固件升级

固件升级的作用主要是针对设备的软件进行升级，升级设备的功能，然后固件升级的操作是要求非常严谨，升级出错会给用户带来非常不好的体验，所以请开发者也谨慎执行此操作，目前 SDK 已经集成了此功能。

升级操作要求在有网络的条件下进行，包括以下 3 个步骤,请务必仔细核对。

具体的升级案例,可参考我司的运行 demo 工程 VpBluetoothSDKDemo 下的 com.timaimee.vpbluetoothsdkdemo.oad.Activity.OadActivity

1.AndroidMainfest.xml 的配置

这个配置已在前面的 Activity&Service 配置进行过说明，主要是配置两个文件,一个 service 和一个 activity,开发者可以直接拷贝 VpBluetoothSDKDemo 下的这两个文件。

2.升级前的判断

a.升级前先进行设备版本校验，升级文件校验

b.两者进行校验后，发送固件升级命令，设备会进入固件升级模式

c.然后 app 找到固件升级模式下的设备时

d.最后可以调用官方升级程序。

checkVersionAndFile()方法已封装以上的所有步骤，开发者只许调用此方法就行。

回调 onCheckSucces 表示设备版本以及升级文件两者校验成功；

回调 findOadDevice 表示找到固件升级模式下的设备,可以调用官方升级程序。

当然有时候可能设备版本以及升级文件已经通过，只是没有发现固件升级模式下的设备，开发者确保设备版本以及升级文件没有问题的情况下，也可以在这之后调用 findOadModelDevice()方法，在回调 findOadDevice 后调用官方升级程序，当然这种情况极少，不建议开发单独调用 findOadModelDevice();

3.调用官方升级程序

OadActivity 下的 startOad()，一般在接口 OnFindOadDeviceListener 的 findOadDevice 方法回调后被调用。

六、其他说明

因项目紧急，目前文档还在完善中，可多查看 Demo 工程以及 API 文档，有疑难问题请咨询我司，给您带来不便，敬请谅解！