FeasyBlue SDK Android 接入文档

Table of contents

[English]

Chapter 1

SDK 接入文档

对应 SDK 版本	编写日期	编写人员
V3.3.1	2024 年 4 月 13 日	常纪刚

1.1 接入方式

解压 SDK，把解压的全部文件拖到工程里。

1.1.1 权限申请

使用 SDK 必须动态获取定位权限，打开手机定位开关与手机蓝牙开关才能正常使用（Android 6.0 以上如果没有动态获取定位权限并打开手机定位开关将扫描不到 BLE 设备，Android 的限制）

AndroidManifest.xml 中需加入以下权限

```
<uses-permission android:name="android.permission.BLUETOOTH_ADMIN" />
<uses-permission android:name="android.permission.BLUETOOTH" />
<uses-permission android:name="android.permission.ACCESS_COARSE_
↪LOCATION" />
<uses-permission android:name="android.permission.ACCESS_FINE_
↪LOCATION" />

// Android 12 需要以下权限
<uses-permission android:name="android.permission.BLUETOOTH_SCAN"/>
```

(continues on next page)

(continued from previous page)

```
<uses-permission android:name="android.permission.BLUETOOTH_CONNECT"/
↪>
```

```
// 需要导入 Kotlin 协程环境
implementation "org.jetbrains.kotlinx:kotlinx-coroutines-core:
↪$coroutines_version"
```

1.1.2 初始化

- BLE 初始化

```
// getInstance(context) 与 initialize() 只需要执行一次，后续其他
Activity 需要使用到 FscBleCentralApi 的地方 直接使用␣
↪FscSppCentralApiImp.getInstance() 即可
FscBleCentralApi sFscBleCentralApi = FscBleCentralApiImp.
↪getInstance(context);
sFscBleCentralApi.initialize();
```

- SPP 初始化

```
// getInstance(context) 与 initialize() 只需要执行一次，后续其他
Activity 需要使用到 FscSppCentralApi 的地方 直接使用␣
↪FscSppCentralApiImp.getInstance() 即可
FscSppCentralApi sFscSppCentralApi = FscSppCentralApiImp.
↪getInstance(context);
sFscSppCentralApi.initialize();
```

1.2 接口统计

1.2.1 BLE 接口

```
/**
* 设置回调
* @param callback 回调
*/
```

(continues on next page)

(continued from previous page)

```
void setCallbacks(FscBleCentralCallbacks callback);

/**
* 获取最后连接设备的 mtu 值
* @return mtu
*/
int getMtu();

/**
* 获取指定设备的 mtu 值
* param address 设备地址
* @return mtu
*/
int getMtu(String address);

/**
* 获取最后连接设备的当前每包最大发送数
* @return
*/
int getMaximumPacketByte();

/**
* 获取指定设备的当前每包最大发送数
* param address 设备地址
* @return
*/
int getMaximumPacketByte(String address);

/**
* 设置最后连接的特征数据
* @param ch            特征值
* @param properties    属性
* @return
*/
boolean setCharacteristic(BluetoothGattCharacteristic ch, int␣
```

(continues on next page)

(continued from previous page)

```
↪properties);

/**
* 设置指定设备的特征数据
* @param address       设备地址
* @param ch            特征值
* @param properties    属性
* @return
*/
boolean setCharacteristic(String address,␣
↪BluetoothGattCharacteristic ch, int properties);

/**
* 读取指定特征值的信息
* @param address   设备地址
* @param ch        特征值
* @return
*/
boolean read(String address, BluetoothGattCharacteristic ch);

/**
* 获取指定设备的服务
* @param address   设备地址
*/
List<BluetoothGattService> getBluetoothGattServices(String address);

/**
* 设置最后连接的设备的 mtu
* @param mtu       mtu 值
*/
@RequiresApi(value = Build.VERSION_CODES.LOLLIPOP)
void requestMtu(int mtu);

/**
* 设置指定设备的 mtu
```

(continues on next page)

(continued from previous page)

```
* @param address   设备地址
* @param mtu       mtu 值
*/
@RequiresApi(value = Build.VERSION_CODES.LOLLIPOP)
void requestMtu(String address, int mtu);

/**
* 绑定设备
*/
void createBond();

/**
* 检查 dfu 文件
* @param dfuFile dfu 文件
* @return {@link DfuFileInfo}
*/
DfuFileInfo checkDfuFile(byte[] dfuFile);
```

1.2.2 SPP 接口


```
/**
* 设置回调
* @param callback 回调
*/
void setCallbacks(FscSppCentralCallbacks callback);

/**
* 打开 SDP 服务，等待设备连接
*/
void openSdpService();

/**
* 关闭 SDP 服务，等待设备连接
*/
void closeSdpService();
```

(continues on next page)

(continued from previous page)

```
/**
* 检查升级文件
* @param dfuFile   dfu 文件
# @return          解析后的信息
*/
DfuFileInfo checkDfuFile(byte[] dfuFile);
```

1.2.3 公用接口


```
/**
* 是否显示日志
* @param isEnableLog
*/
void isShowLog(boolean isEnableLog);

/**
* 初始化
* @return {@link boolean} 初始化结果
*/
boolean initialize();

/**
* 是否已开启蓝牙
* @return {@link boolean} 蓝牙开关状态
*/
boolean isEnabled();

/**
* 删除指定设备配对记录
* @return {@link boolean} 删除是否成功
*/
boolean clearDevice(String address);

/**
```

(continues on next page)

(continued from previous page)

```
* 连接
* @param address 设备地址
*/
void connect(String address);

/**
* 参数修改连接（AT 指令模式）
* @param address 设备地址
*/
void connectToModify(String address);

/**
* 空中升级连接
* @param address 设备地址
* @param dfuFile 升级文件
* @param reset   是否恢复出厂设置
*/
void connectToOTAWithFactory(String address, byte[] dfuFile, boolean
↪reset);

/**
* 断开最后连接的设备
*/
void disconnect();

/**
* 断开指定设备连接
* @param address
*/
void disconnect(String address);

/**
* 获取已绑定的设备
* @return {@link FscDevice}
*/
```

(continues on next page)

(continued from previous page)

```
List<FscDevice> getBondDevices();

/**
* 开始扫描
*/
void startScan();

/**
* 停止扫描
*/
void stopScan();

/**
* 是否有设备已连接
* @return
*/
boolean isConnected();

/**
* 通过设备地址查询连接状态
* @param address 设备地址
* @return
*/
boolean isConnected(String address);

/**
* 给最后连接的设备发送数据
* @param data          发送的数据
* @param sendInterval  发送间隔
* @return
*/
boolean send(String data, Long sendInterval);

/**
* 向指定设备发送信息
```

(continues on next page)

(continued from previous page)

```
* @param address       设备地址
* @param data          发送的数据
* @param sendInterval  发送间隔
* @return
*/
boolean send(String address, String data, Long sendInterval);

/**
* 给最后连接的设备发送数据
* @param packet        发送的数据
* @param sendInterval  发送间隔
* @return
*/
boolean send(byte[] packet, Long sendInterval);

/**
* 向指定设备发送信息
* @param address       设备地址
* @param data          发送的数据
* @param sendInterval  发送间隔
* @return
*/
boolean send(String address, byte[] packet, Long sendInterval);

/**
* 生成指定大小的测试文件发送给最后连接的设备
* @param size          测试文件大小
* @param sendInterval  发送间隔
* @return
*/
boolean sendFile(int size, Long sendInterval);

/**
* 生成指定大小的测试文件发送给指定设备
* @param size          测试文件大小
```

(continues on next page)

(continued from previous page)

```
* @param sendInterval  发送间隔
* @return
*/
boolean sendFile(String address, int size, Long sendInterval);

/**
* 发送文件给最后连接的设备
* @param byteArray          测试文件大小
* @param sendInterval  发送间隔
* @return
*/
boolean sendFile(byte[] byteArray, Long sendInterval);

/**
* 发送文件给指定设备
* @param byteArray         测试文件大小
* @param sendInterval      发送间隔
* @return
*/
boolean sendFile(String address, byte[] byteArray, Long
↪sendInterval);

/**
* 给最后连接的设备发送命令
* @param command   命令集合
*/
void sendATCommand(Set<String> command);

/**
* 给指定设备发送命令
* @param address   设备地址
* @param command   命令集合
*/
void sendATCommand(String address, Set<String> command);
```

(continues on next page)

(continued from previous page)

```
/**
* 停止最后连接的设备发送数据
*/
void stopSend();

/**
* 停止指定设备发送数据
* @param address 设备地址
*/
void stopSend(String address);
```

1.3 API 使用

1.3.1 搜索

- BLE 搜索方法

```
FscBleCentralApi.start();
```

- BLE 搜索例子

```
FscBleCentralApi sFscBleCentralApi = FscBleCentralApiImp.
↪getInstance();
sFscBleCentralApi.setCallbacks(new FscBleCentralCallbacksImp(){
  @Override
  public void blePeripheralFound(FscDevice device, int rssi, byte[]␣
↪record) {
      // 扫描到设备
  }
});
sFscBleCentralApi.startScan();
```

- BLE 停止搜索方法

```
FscBleCentralApi.stopScan();
```

- SPP 搜索方法

```
FscSppCentralApi.start();
```

- SPP 搜索例子

```
FscSppCentralApi sFscSppCentralApi = FscSppCentralApiImp.
↪getInstance();
sFscSppCentralApi.setCallbacks(new FscSppCentralCallbacksImp(){
  @Override
  public void sppPeripheralFound(FscDevice sppDevice, int rssi) {
      // 发现设备
  }
});
sFscSppCentralApi.startScan();
```

- SPP 停止搜索方法

```
FscSppCentralApi.stopScan();
```

1.3.2 连接

- BLE 连接方法

```
FscBleCentralApi sFscBleCentralApi = FscBleCentralApiImp.
↪getInstance();
sFscBleCentralApi.setCallbacks(new FscBleCentralCallbacksImp(){
  @Override
  public void blePeripheralConnected(BluetoothGatt gatt, String␣
↪address)  {
      // 连接成功
  }
});
sFscBleCentralApi.connect();
```

- SPP 连接方法

```
FscSppCentralApi sFscSppCentralApi = FscSppCentralApiImp.
↪getInstance();
sFscSppCentralApi.setCallbacks(new FscSppCentralCallbacksImp(){
```

(continues on next page)

(continued from previous page)

```
  @Override
  public void sppPeripheralConnected(BluetoothDevice device) {
      // 连接成功
  }
});
sFscSppCentralApi.connect();
```

1.3.3 断开连接

- BLE 断开连接方法

```
FscBleCentralApi sFscBleCentralApi = FscBleCentralApiImp.
↪getInstance();
sFscBleCentralApi.setCallbacks(new FscBleCentralCallbacksImp(){
  @Override
  public void blePeripheralDisconnected(BluetoothGatt gatt, String␣
↪address) {
      // 断开连接
  }
});
sFscBleCentralApi.disconnect();
```

- SPP 断开连接方法

```
FscSppCentralApi sFscSppCentralApi = FscSppCentralApiImp.
↪getInstance();
sFscSppCentralApi.setCallbacks(new FscSppCentralCallbacksImp(){
  @Override
  public void sppPeripheralDisconnected(String address) {
      // 断开连接
  }
});
sFscSppCentralApiImp.disconnect();
```

1.3.4 回调

- BLE 回调方法

```
/**
* 扫描到设备
* @param device          扫描出的设备
* @param rssi              信号值
* @param scanRecord      扫描到的广播信息
*/
void blePeripheralFound(FscDevice device, int rssi, byte[]␣
↪scanRecord);

/**
* 发送设备
* @param gatt         BluetoothGatt 对象
* @param address      设备地址
*/
void blePeripheralConnected(BluetoothGatt gatt, String address);

/**
* 断开连接
* @param gatt         BluetoothGatt 对象
* @param address      设备地址
*/
void blePeripheralDisconnected(BluetoothGatt gatt, String address);

/**
* 发现服务
* @param gatt         BluetoothGatt 对象
* @param address      设备地址
* @param services     服务
*/
void servicesFound(BluetoothGatt gatt, String address, List
↪<BluetoothGattService> services);

/**
* 发现服务
* @param gatt         BluetoothGatt 对象
* @param address      设备地址
```

(continues on next page)

(continued from previous page)

```
* @param service      服务
* @param characteristic 特征值
*/
void characteristicForService(BluetoothGatt gatt, String address,␣
↪BluetoothGattService service, BluetoothGattCharacteristic␣
↪characteristic);

/**
* 读特征值信息
* @param address      设备地址
* @param ch           特征值
* @param strValue     读到的数据
* @param hexString    读到的十六进制数据
* @param rawValue     原数据
*/
void readResponse(String address, BluetoothGattCharacteristic ch,␣
↪String strValue, String hexString, byte[] rawValue);

/**
* mtu 变更
* @param mtu          请求后的 mtu 值
* @param status       请求 mtu 的状态
*/
void bleMtuChanged(int mtu, int status);
```

- SPP 回调方法

```
/**
* 扫描到设备
* @param sppDevice 设备地址
*/
void sppPeripheralFound(FscDevice sppDevice, int rssi);

/**
* 连接成功
* @param device 连接成功的设备
```

(continues on next page)

(continued from previous page)

```
*/
void sppPeripheralConnected(BluetoothDevice device);

/**
* 断开连接
* @param address 设备地址
*/
void sppPeripheralDisconnected(String address);

/**
* 发送数据
* @param address        设备地址
* @param strValue       发送的数据
* @param hexString      发送的十六进制数据
* @param data           原数据
*/
void packetSend(String address, String strValue, String hexString,
↪byte[] data);
```

- 相同回调方法

```
/**
* 开始扫描回调
*/
void startScan();

/**
* 停止扫描
*/
void stopScan();

/**
* 收到数据
* @param address 设备地址
* @param strValue 字符串
* @param dataHexString 十六进制
```

(continues on next page)

(continued from previous page)

```
* @param data 源数据
*/
void packetReceived(String address, String strValue, String
↪dataHexString, byte[] data);

/**
* 发送数据
* @param address 设备地址
* @param strValue 字符串
* @param data 源数据
*/
void packetSend(String address, String strValue, byte[] data);

/**
* 发送文件进度
* @param address 设备地址
* @param percentage 进度
* @param data 源数据
*/
void sendPacketProgress(String address, int percentage, byte[] data);

/**
* OTA 升级进度
* @param address 设备地址
* @param percentage 进度
* @param status 状态
*/
void otaProgressUpdate(String address, int percentage, int status);

/**
* AT 指令模式通讯回调
* @param command 发送的命令
* @param parameter 收到的回复
* @param type 类型
* @param status 状态
```

(continues on next page)

(continued from previous page)

```
*/
void atCommandCallBack(String command, String parameter, int type,␣
↪int status);

/**
* AT 指令发送结束时触发
*/
void endATCommand();

/**
* 开始发送 AT 指令时触发
*/
void startATCommand();
```

1.3.5 发送数据

- BLE 发送数据方法

```
FscBleCentralApi sFscBleCentralApi = FscBleCentralApiImp.
↪getInstance();
sFscBleCentralApiImp.send("abc");
```

- SPP 发送数据方法

```
FscSppCentralApi sFscSppCentralApi = FscSppCentralApiImp.
↪getInstance();
sFscSppCentralApiImp.send("abc");
```

1.3.6 空中升级

- BLE 空中升级方法

```
FscBleCentralApi sFscBleCentralApi = FscBleCentralApiImp.
↪getInstance();

sFscBleCentralApi.setCallbacks(new FscBleCentralCallbacksImp(){
```

(continues on next page)

(continued from previous page)

```
  public void otaProgressUpdate(address: String, percentage: Int,␣
↪status: Int) {
      // 升级进度
  }
});
sFscBleCentralApi.connectToOTAWithFactory(
                  fscDevice.getAddress(),
                  dfuByte,
                  reset
              );
```

- SPP 空中升级方法

```
FscSppCentralApi sFscSppCentralApi = FscSppCentralApiImp.
↪getInstance();
sFscSppCentralApi.setCallbacks(new FscSppCentralCallbacksImp(){
  public void otaProgressUpdate(address: String, percentage: Int,␣
↪status: Int) {
      // 升级进度
  }
});
sFscSppCentralApi.connectToOTAWithFactory(
                  fscDevice.getAddress(),
                  dfuByte,
                  reset
              );
```

Chapter 2

附件

2.1 下载 PDF 版本

下载 PDF 版本