[TOC]

版本

版本号	变更内容	作者	时间
1.0.0	- 初始文档；	Xerxes	2023.08.25

SDK接口说明

1. 概述

本文档旨在提供xBeaconSDK接口的详细说明，以便开发人员能够准确地使用xBeaconSDK进行应用程序开发。xBeaconSDK（xBeacon Software Development Kit）是一组集成蓝牙扫描连接和对Beacon设备进行相应的配置的SDK。

2. 安装和配置

在开始使用SDK之前，您需要按照以下步骤进行安装和配置：

1. 您可能需要在工程根目录下的gradle文件中引入远程仓库地址：

```
google()
mavenCentral()
maven { url "https://jitpack.io" }
```

1. 在您的app模块 下的build.gradle文件中引入以下远程库和添加本地库的引用：

```
implementation  fileTree(include: ['*.jar', '*.aar'], dir: 'libs')
implementation 'no.nordicsemi.android.support.v18:scanner:1.6.0'
implementation 'no.nordicsemi.android:log:2.3.0'
implementation 'no.nordicsemi.android:ble-livedata:2.6.0'
```

2.1 下载SDK

请联系我们: shenzhen@freqchina.com

2.2 解压安装包

将您的我们提供的SDK `xBeacon.aar` 文件放入app 模块的 `libs` 文件夹下

2.3 导入SDK到项目

当您文件放好,gradle参数配置完成后,在Android studio点击sync 同步即可成功导入,您也可以参考我们的DEMO进行了解

2.4 配置SDK参数

根据您的应用程序需求，配置SDK参数。这些参数可能包括SDK初始化,过滤参数配置,连接参数配置,密码配置和其他相关设置。请参考SDK文档获取详细的配置说明。并且在开始接入接口之前首先完成SDK初始化：

你可以在你的application 或者 其他还未开始使用SDK之前的页面完成初始化：

`XtManager.initialize(this)`

3. SDK接口列表

调用所有接口均采用 `XtManager.getInstance()` 来引入

以下是SDK提供的主要接口列表：

3.1 设置扫描过滤条件

此接口用于设置扫描过滤条件。具体方法和参数如下：

```
fun setScanFilters(xtScanFilters: XtScanFilter)
```

参数说明：

- xtScanFilters：过滤参数对象，可以设置是否打开过滤以及名字过滤，信号过滤，设备类型过滤.

3.2 扫描设备

此接口用于实现BLE设备扫描。具体方法和参数如下：

```
fun startScan()
```

参数说明：

- 无参数

3.3 停止扫描

此接口用于停止BLE的扫描。具体方法和参数如下：

```
fun stopScan()
```

参数说明：

- 无参数

3.4 蓝牙扫描结果监听

此接口用于监听BLE设备的扫描结果。具体方法和参数如下：

```
fun setBleScanListener(listener: BleScanResultListener)
```

参数说明：

- listener 是扫描设备的结果的监听接口,此接口包含如下内容：

```
    /**
     * BLE scan result listener interface for receiving BLE scan results
     */
    interface BleScanResultListener {
        /**
         * Called when scan results are available.
         */
        fun onScanResult(scanedBluetoothDevice: MutableList<ScanedBlueto
    }
```

3.5 连接设备

此接口用于连接BLE设备。具体方法和参数如下：

```
connect(device: ScanedBluetoothDevice)
```

参数说明：

- device 是通过扫描接口返回的设备对象

3.6 断接设备

此接口用于断接BLE设备。具体方法和参数如下：

```
fun disconnect()
```

参数说明：

- 无参数

3.7 断接设备

此接口用于断接BLE设备。具体方法和参数如下：

```
fun disconnect()
```

参数说明：

- 无参数

3.8 设备连接状态

此接口用于判断BLE设备是否连接。具体方法和参数如下：

```
fun isConnected(): Boolean
```

参数说明：

- 无参数

3.9 设置设备的连接密码

此接口用于设置设备的连接密码。具体方法和参数如下：

```
fun modifyPsd(psd: String, isEnable: Boolean)
```

参数说明：

- psd 连接的密码,要求6位
- isEnable 是否启用自己配置的密码连接设备

3.10 读取数据

此接口用于读取设备的各种数据。具体方法和参数如下：

```
fun readData(opcode: Int)
```

参数说明：

- opcode 您要读取的数据类型： ```java // UUID var OPCODE_UUID = 0X23001

  // BEACON parameter MAJOR var OPCODE_MAJOR = 0X23002

  // BEACON parameter MINOR var OPCODE_MINOR = 0X23003

  // BEACON parameter transmit power var OPCODE_POWER = 0X23004

  // BEACON parameter advertising interval var OPCODE_GAP = 0X23005

  // BEACON parameter one-meter signal value var OPCODE_MRSSI = 0X23006

  // BEACON device battery level var OPCODE_BATTERY = 0X23007

  // BEACON all parameters var OPCODE_ALL = 0X23008 ```

3.11 配置数据

以下接口用于配置设备的各种数据。具体方法和参数如下：

```
/**
 * Set the major parameter value.
 */
fun setMajor(major: Int) {}

/**
 * Set the minor parameter value.
 */
fun setMinor(minor: Int) {}

/**
 * Set the UUID parameter value.
 */
fun setUUID(uuid: String) {}

/**
 * Set the mRssi parameter value.
 */
fun setMRssi(mRssi: Int) {}

/**
 * Configure the password.
 */
fun configPsd(password: String) {}

/**
 * Set the power parameter value.
 */
fun setPower(power: Int) {}

/**
 * Set the broadcast interval parameter value.
 */
fun setGap(gap: Int) {}
```

3.12 其他数据监听

1. `fun setBleStateListener(listener: BleStateListener)` 监听蓝牙的状态信息，具体如下：

```
/**
     * BLE state listener interface for receiving BLE state changes.
     */
    interface BleStateListener {
        /**
         * Called when the BLE connection is disconnected.
         */
        fun onDisConnected()

        /**
         * Called when the BLE connection is established.
         */
        fun onConnected()

        /**
         * Called when the BLE connection is established and ready, with
         */
        fun onReady()
    }
```

1. fun setBleResultListener(listener: BleResultListener) 写数据结果监听：

```
/**
     * BLE result listener interface for receiving BLE configuration resu
     */
    interface BleResultListener {
        /**
         * Called when major configuration is completed.
         */
        fun onConfigMajorResult(isSuccess: Boolean)

        /**
         * Called when minor configuration is completed.
         */
        fun onConfigMinorResult(isSuccess: Boolean)

        /**
         * Called when UUIDconfiguration is completed.
         */
        fun onConfigUuidResult(isSuccess: Boolean)

        /**
         * Called when gap configuration is completed.
         */
        fun onConfigGapResult(isSuccess: Boolean)

        /**
         * Called when power configuration is completed.
         */
        fun onConfigPowerResult(isSuccess: Boolean)

        /**
         * Called when mRssi configuration is completed.
         */
        fun onConfigMRssiResult(isSuccess: Boolean)

        /**
         * Called when password configuration is completed.
         */
        fun onConfigPsd(isSuccess: Boolean)
    }
```

1. fun setBleDataReceiveListener(listener: BleDataReceiveListener) 读数据返回值：

```
/**
     * BLE data receive listener interface for receiving BLE data.
     */
    interface BleDataReceiveListener {
        /**
         * Called when data is received with an opcode and integer data.
         */
        fun receiveData(opcode: Int, data: Int?)

        /**
         * Called when data is received with an opcode and string data.
         */
        fun receiveData(opcode: Int, data: String?)
    }
```

4. 使用示例

以下是使用SDK接口的示例代码：

```
//Scan related configuration


class MainActivity : AppCompatActivity(), XtManager.BleStateListener {

private fun initScanDev() {
        //scan filter
        var xtScanFilter = XtScanFilter
            .setFilterRssi(-48)
            .enableFilterDevice(true)
//            .setFilterDevType(DeviceTypeOp.CARD_TYPE_KC_B1)
//            .setFilterName("AABB")

        //Set up scan filters
        XtManager.getInstance().setScanFilters(xtScanFilter)

        //start scanning
        XtManager.getInstance().startScan()

        //scan result monitoring
        XtManager.getInstance().setBleScanListener(object : XtManager.Bl
            override fun onScanResult(scanedBluetoothDevice: MutableList
                //To process the scan result, the device that needs to b
                // XtManager.getInstance().connect(adapter.getItem(posit
            }
        })

        //Monitor Bluetooth status
        XtManager.getInstance().setBleStateListener(this@MainActivity)
    }

    override fun onDisConnected() {

    }

    override fun onConnected() {

    }

    /**
     * The connection is successful and the password is correct
     */
    override fun onReady() {
       // Set parameters according to the setting interface
    }
}
```

5. 错误处理

在使用SDK接口时，可能会遇到一些错误情况。SDK提供了错误LOG信息显示，以便开发人员能够及时处理错误并采取适当的措施。错误LOG信息的标题为：“XTLOG:”

6. 参考文档

在开发过程中，您可能需要查阅SDK的更多详细信息和示例代码。请参考以下文档：

- SDK的使用示例DEMO

7. 常见问题解答

在开发和使用SDK过程中，您可能会遇到一些常见问题。以上SDK示例文档和DEMO，如有其他问题可以联系我们。