[TOC]

Version

Version number	Change content	Author	Time
1.0.0	- Initial documentation;	Xerxes	2023.08.25

SDK interface description

1 Overview

This document aims to provide a detailed description of the xBeaconSDK interface so that developers can accurately use xBeaconSDK for application development. xBeaconSDK (xBeacon Software Development Kit) is a set of SDKs that integrate Bluetooth scanning connections and configure Beacon devices accordingly.

2. Installation and configuration

Before starting to use the SDK, you need to follow the steps below to install and configure it:

1. You may need to introduce the remote warehouse address in the gradle file under the project root directory:

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

1. Introduce the following remote libraries and add references to local libraries in the build.gradle file under your app module:

```
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 Download the SDK

Please contact us: shenzhen@freqchina.com

2.2 Place the installation package

Put your SDK `xBeacon.aar` file provided by us into the `libs` folder of the app module

2.3 Import SDK to project

After your file is placed and the gradle parameter configuration is complete, click sync in Android studio to successfully import. You can also refer to our DEMO to learn more

2.4 Configure SDK parameters

Configure SDK parameters according to your application requirements. These parameters may include SDK initialization, filtering parameter configuration, connection parameter configuration, password configuration and other related settings. Please refer to the SDK documentation for detailed configuration instructions. And complete the SDK initialization first before starting to access the interface:

You can complete the initialization in your application or other pages before starting to use the SDK:

```
XtManager.initialize(this)
```

3. SDK interface list

Call all interfaces using `XtManager.getInstance()` to import

The following is a list of the main interfaces provided by the SDK:

3.1 Set scanning filter conditions

This interface is used to set scanning filter conditions. The specific methods and parameters are as follows:

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

Parameter Description:

- xtScanFilters: filter parameter object, you can set whether to open filter and name filter, signal filter, device type filter.

3.2 Scanning Devices

This interface is used to implement BLE device scanning. The specific methods and parameters are as follows:

```
fun startScan()
```

Parameter Description:

- no parameters

3.3 Stop scanning

This interface is used to stop BLE scanning. The specific methods and parameters are as follows:

```
fun stopScan()
```

Parameter Description:

- no parameters

3.4 Bluetooth scan result monitoring

This interface is used to monitor the scan results of BLE devices. The specific methods and parameters are as follows:

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

Parameter Description:

- listener is the listening interface for the result of scanning the device. This interface contains the following content:

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

3.5 Connecting Devices

This interface is used to connect to BLE devices. The specific methods and parameters are as follows:

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

Parameter Description:

- device is the device object returned by the scan interface

3.6 Disconnect device

This interface is used to disconnect BLE devices. The specific methods and parameters are as follows:

```
fun disconnect()
```

Parameter Description:

- no parameters

3.7 Disconnect device

This interface is used to disconnect BLE devices. The specific methods and parameters are as follows:

```
fun disconnect()
```

Parameter Description:

- no parameters

3.8 Device Connection Status

This interface is used to determine whether the BLE device is connected. The specific methods and parameters are as follows:

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

Parameter Description:

- no parameters

3.9 Set the connection password of the device

This interface is used to set the connection password of the device. The specific methods and parameters are as follows:

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

Parameter Description:

- The password for the psd connection requires 6 characters
- isEnable Whether to enable your own configured password to connect to the device

3.10 Read data

This interface is used to read various data of the device. The specific methods and parameters are as follows:

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

Parameter Description:

- opcode The type of data you want to read: ```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 Configuration data

The following interfaces are used to configure various data of the device. The specific methods and parameters are as follows:

```
/**
 * 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 Other data monitoring

1. `fun setBleStateListener(listener: BleStateListener)` Monitor the status information of Bluetooth, as follows:

```
/**
     * 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) Write data result monitoring:

```
/**
     * 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) Read data return value:

```
/**
     * 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. Example of use

The following is the sample code using the SDK interface:

```
//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. Error Handling

When using the SDK interface, you may encounter some error conditions. The SDK provides error log information display so that developers can handle errors in time and take appropriate measures. The title of the error LOG information is: "XTLOG:"

6. Reference Documentation

During development, you may need to consult the SDK for more details and sample code. Please refer to the following documents:

- SDK usage example DEMO

7. Frequently Asked Questions

During the process of developing and using the SDK, you may encounter some common problems. The above SDK sample documents and DEMO, if you have any other questions, please contact us.