



# ChipletRing APP SDK 说明书

## Android

|        |          |       |             |
|--------|----------|-------|-------------|
| 文档版本号: | 1.3.7    | 文档编号: |             |
| 文档密级:  |          | 归属部门: | Chiplet 研发部 |
| 产品名:   | BCL_603M | 归属项目: |             |
| 编写人:   | 郑玉虎      | 编写日期: | 2023.10.19  |

## 修订记录:

| 版本号   | 修订人 | 修订日期       | 修订描述                                |
|-------|-----|------------|-------------------------------------|
| 1.0.0 | 郑玉虎 | 2023.10.19 | 修改导航目录                              |
| 1.0.1 | 郑玉虎 | 2023.10.26 | 修改集成和初始化 SDK 的描述, 修改部分演示代码          |
| 1.0.2 | 郑玉虎 | 2023.10.31 | 补全代码, 加入问题解答                        |
| 1.0.3 | 郑玉虎 | 2023.11.23 | 更新问题解答, 修改读取数据的参数描述                 |
| 1.0.4 | 郑玉虎 | 2023.11.29 | 新增 Q&A, 作为后续简单问题汇总地                 |
| 1.1.1 | 郑玉虎 | 2023.12.26 | 1.新增蓝牙扫描时的筛选方法<br>2.测量心率、测量血氧增加上传数据 |
| 1.1.2 | 郑玉虎 | 2024.01.02 | 增加波形图数据分析                           |
| 1.1.3 | 郑玉虎 | 2024.04.19 | 增加心率血氧 PPG 原始数据解析                   |
| 1.3   | 郑玉虎 | 2024.05.20 | 增加对采集周期的详细说明                        |
| 1.3.1 | 郑玉虎 | 2024.06.12 | 增加对步数的详细说明                          |
| 1.3.2 | 郑玉虎 | 2024.07.09 | 增加睡眠逻辑流程图                           |
| 1.3.3 | 郑玉虎 | 2024.07.17 | 增加血压算法的接口                           |
| 1.3.4 | 郑玉虎 | 2024.08.15 | 修改对同步时间和读取时间的描述                     |
| 1.3.5 | 郑玉虎 | 2024.11.04 | 增加设置和读取蓝牙名称、实时血压接口, 增加长连接<br>相关功能   |
| 1.3.6 | 郑玉虎 | 2024.11.22 | 增加心率、血氧停止的接口说明                      |
| 1.3.7 | 郑玉虎 | 2024.11.25 | 增加一键获取状态接口, 新增广播 0018, 语音录制         |

## 目 录

|   |    |
|---|----|
| 一、 文档简介 .....   | 6  |
| 1、 文档目的 .....   | 6  |
| 2、 适用范围 .....   | 6  |
| 3、 功能介绍 .....   | 6  |
| 二、 快速入门概览 .....   | 7  |
| 1、 前置条件 .....   | 7  |
| 2、 使用流程 .....   | 7  |
| 3、 流程图 .....  | 8  |
| 三、 集成 ChipletRing APP SDK .....                                   | 9  |
| 1、 集成 ChipletRing APP SDK .....                                   | 9  |
| 1.1 集成方式 .....  | 9  |
| 1.1.1 获取到 ChipletRing APP SDK 的 jar 包文件 .....                     | 9  |
| 1.1.2 将 jar 包放在 libs 目录下 .....                                    | 9  |
| 1.1.3 右键设置为 Add as Library(添加为类库) .....                           | 9  |
| 1.1.4 配置所需权限，如需存储以及其他权限可自行配置，牵扯到动态权限处，需要做<br>相关处理 .....           | 9  |
| 2、 初始化 ChipletRing APP SDK .....                                  | 10 |
| 2.1.1 在 Application 的 onCreate 方法中进行初始化 .....                     | 10 |
| 2.1.2 在 BaseActivity 类中，启用监听，该监听用于监听蓝牙连接状态以及戒指基础指<br>令反馈的数据 ..... | 10 |
| 3、 使用 ChipletRing APP SDK .....                                   | 11 |
| 3.1 蓝牙操作（BLEUtils） .....  | 11 |
| 3.1.1 搜索设备 .....  | 11 |
| 3.1.2 停止搜索 .....  | 12 |
| 3.1.3 连接设备 .....  | 12 |
| 3.1.4 断开蓝牙 .....  | 13 |
| 3.2 指令功能（LmAPI） .....   | 13 |

|                               |    |
|-------------------------------|----|
| 3.2.1 同步时间 .....              | 13 |
| 3.2.2 读取时间 .....              | 14 |
| 3.2.3 版本信息 .....              | 14 |
| 3.2.4 电池电量 .....              | 15 |
| 3.2.5 读取步数 .....              | 15 |
| 3.2.6 清除步数 .....              | 16 |
| 3.2.7 恢复出厂设置 .....            | 16 |
| 3.2.8 采集周期设置 .....            | 16 |
| 3.2.9 采集周期读取 .....            | 17 |
| 3.2.10 测量心率 .....             | 17 |
| 3.2.11 测量血氧 .....             | 19 |
| 3.2.12 测量温度 .....             | 20 |
| 3.2.13 历史记录管理 .....           | 20 |
| 3.2.14 清空历史数据 .....           | 21 |
| 3.2.15 血压测试算法 .....           | 22 |
| 3.2.16 实时 PPG 血压测量 .....      | 22 |
| 3.2.17 实时 PPG 血压停止采集 .....    | 23 |
| 3.2.18 设置蓝牙名称 .....           | 23 |
| 3.2.19 获取蓝牙名称 .....           | 24 |
| 3.2.20 心率测量停止 .....           | 24 |
| 3.2.21 血氧测量停止 .....           | 24 |
| 3.2.22 一键获取状态 .....           | 25 |
| 3.2.23 语音录制 .....             | 25 |
| 3.3 固件升级 (OTA) .....          | 26 |
| 3.3.1 检查版本 .....              | 26 |
| 3.3.2 开始升级 .....              | 27 |
| 3.4 数据库相关 (DataApi) .....     | 28 |
| 3.4.1 查询历史记录 .....            | 28 |
| 3.4.2 清空历史数据 .....            | 29 |
| 3.5 逻辑算法相关 (LogicalApi) ..... | 30 |

|                                       |           |
|---------------------------------------|-----------|
| 3.5.1 计算距离、卡路里 .....                  | 30        |
| 3.5.2 计算睡眠数据 .....                    | 30        |
| <b>四、 其他 .....</b>                    | <b>32</b> |
| 1、 筛选相关 .....                         | 32        |
| 1.1 广播数据单元 .....                      | 33        |
| 1.1.1 Raw data: .....                 | 33        |
| 1.1.2 Details: .....                  | 34        |
| 1.2 应用 .....                          | 34        |
| 1.2.1 广播最新版本 .....                    | 34        |
| 2、 可能会遇到的问题 .....                     | 35        |
| 2.1 版本相关 .....                        | 35        |
| 2.2 Gradle 4.0 以上导致 Xpopup 无法使用 ..... | 35        |
| 2.3 不需要界面 .....                       | 35        |
| 2.4 OTA 类引用未找到 .....                  | 35        |
| 3、 硬件算法逻辑或固件相关 .....                  | 36        |
| 3.1 戒指相关 .....                        | 36        |
| 3.2 算法相关 .....                        | 36        |
| 3.3 睡眠逻辑图 .....                       | 37        |
| 3.4 午睡逻辑图 .....                       | 38        |
| 4、 Q&A .....                          | 38        |

## 一、 文档简介

### 1、 文档目的

为方便 Android 端 APP 与戒指通讯进行二次开发，特对通讯协议进行封装，以达到简洁明了，让开发者不需要关注与戒指通讯层，专注业务逻辑交互层面开发。

### 2、 适用范围

本 SDK 基于安卓原生开发，最终提供为 jar 包库。可用于 Android 环境下使用。

### 3、 功能介绍

| 功能模块   | 说明   | 相关文档 |
|--------|--|------|
| 蓝牙基础模块 | <ul style="list-style-type: none"><li>● 蓝牙的开关操作</li><li>● 蓝牙的搜索链接操作</li><li>● 蓝牙的数据写入监听操作</li></ul>  |      |
| 通讯协议模块 | <ul style="list-style-type: none"><li>● 时间管理</li><li>● 版本号管理</li><li>● 电池管理</li><li>● 心率测量</li><li>● 血氧测量</li><li>● 温度测量</li><li>● 计步管理</li><li>● 历史记录管理</li><li>● 系统设置</li><li>● 日志管理</li></ul> |      |

## 二、快速入门概览

### 1、前置条件

- Android系统环境，系统版本 $\geq 5.0$
- 必须支持蓝牙5.0
- 使用语言须可以调用jar包

### 2、使用流程

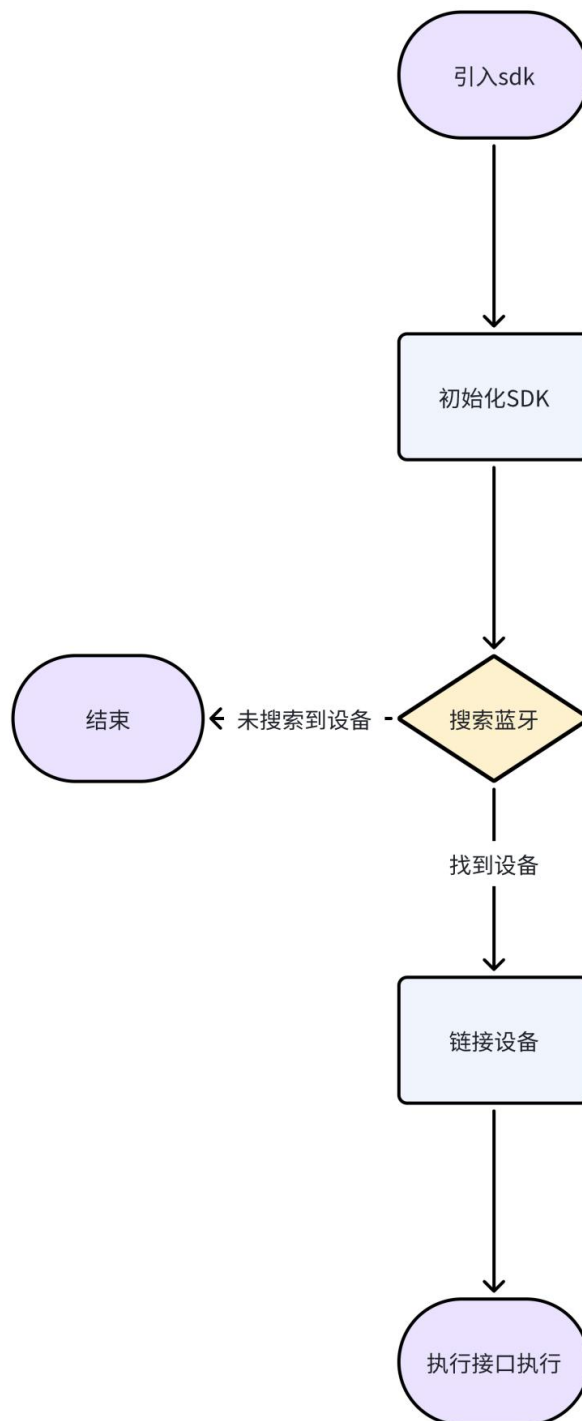
按以下流程使用 SDK:

第一步: 集成 SDK

第二步: 初始化 SDK

第三步: 使用 SDK

### 3、 流程图





### 三、集成 ChipletRing APP SDK

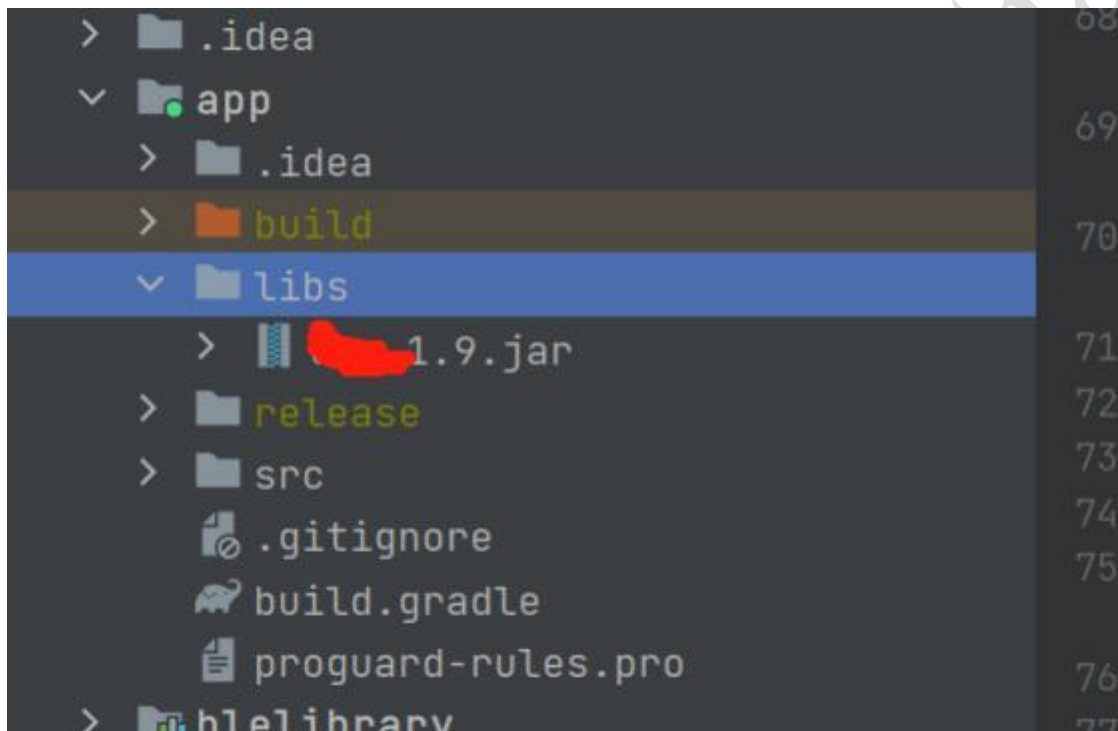
#### 1、集成 ChipletRing APP SDK

##### 1.1 集成方式

###### 1.1.1 获取到 ChipletRing APP SDK 的 jar 包文件

###### 1.1.2 将 jar 包放在 libs 目录下

###### 1.1.3 右键设置为 Add as Library(添加为类库)



###### 1.1.4 配置所需权限，如需存储以及其他权限可自行配置，牵扯到动态权限处，需要做相关处理

在 Manifest.xml 中加入以下代码

```
1. <uses-permission android:name="android.permission.BLUETOOTH_ADVERTISE" />
2. <uses-permission android:name="android.permission.BLUETOOTH_CONNECT" />
3. <uses-permission android:name="android.permission.BLUETOOTH_SCAN" />
4. <uses-permission android:name="android.permission.ACCESS_COARSE_LOCATION" />
5. <uses-permission android:name="android.permission.ACCESS_FINE_LOCATION" />
```

## 2、 初始化 ChipletRing APP SDK

### 2.1.1 在 Application 的 onCreate 方法中进行初始化

```
1. LmAPI.init(this);  
2. LmAPI.setDebug(true);
```

### 2.1.2 在 BaseActivity 类中，启用监听，该监听用于监听蓝牙连接状态以及戒指基础指令反馈的数据

```
1. LmAPI.addWLSCmdListener(this, this);  
2. // 监视蓝牙设备与APP连接的状态  
3. IntentFilter intentFilter = new IntentFilter();  
4. intentFilter.addAction(BluetoothAdapter.ACTION_STATE_CHANGED);  
5. intentFilter.addAction(BluetoothDevice.ACTION_ACL_DISCONNECTED);  
6. intentFilter.addAction(BluetoothDevice.ACTION_ACL_CONNECTED);  
7. registerReceiver(broadcastReceiver,intentFilter);  
8.  
9. //使用蓝牙之前，先申请去权限  
10. if (Build.VERSION.SDK_INT >= Build.VERSION_CODES.S) {  
11.     if (!checkPermissions(new String[]{Manifest.permission.ACCESS_COARSE_L  
LOCATION, Manifest.permission.ACCESS_FINE_LOCATION, Manifest.permission.BLUE  
TOOTH_SCAN, Manifest.permission.BLUETOOTH_CONNECT, Manifest.permission.BLUE  
TOOTH_ADVERTISE})) {  
12.         new XPopup.Builder(this).asConfirm(getRsString(R.string.hint), get  
String(R.string.localtion_auth),  
13.             new OnConfirmListener() {  
14.                 @Override  
15.                 public void onConfirm() {  
16.                     requestPermission(new String[]{Manifest.permission  
.ACCESS_COARSE_LOCATION, Manifest.permission.ACCESS_FINE_LOCATION, Manifest  
.permission.BLUETOOTH_SCAN, Manifest.permission.BLUETOOTH_CONNECT, Manifest  
.permission.BLUETOOTH_ADVERTISE}, 100);  
17.                 }  
18.             }).show();  
19.         return;  
20.     }  
21.  
22. } else {  
23.     if (!checkPermissions(new String[]{Manifest.permission.ACCESS_COARSE_L  
LOCATION, Manifest.permission.ACCESS_FINE_LOCATION})) {  
24.         new XPopup.Builder(this).asConfirm(getRsString(R.string.hint), get  
String(R.string.localtion_auth),
```

```
25.         new OnConfirmListener() {
26.             @Override
27.             public void onConfirm() {
28.                 requestPermission(new String[]{Manifest.permission
29.                     .ACCESS_COARSE_LOCATION, Manifest.permission.ACCESS_FINE_LOCATION}, 100);
30.             }
31.         }.show();
32.     return;
33. }
```

### 3、 使用 ChipletRing APP SDK

#### 3.1 蓝牙操作（BLEUtils）

此类是使用蓝牙搜索、连接、断开的公共类，统一由 IResponseListener 接口反馈。

##### 3.1.1 搜索设备

接口功能：开启蓝牙搜索功能，搜索周围的蓝牙设备。

接口声明：

```
1. BLEUtils.startLeScan(Context context, BluetoothAdapter.LeScanCallback leScanCallback);
```

参数说明：context：上下文      leScanCallback：蓝牙搜索的回调

返回值（void onLeScan(BluetoothDevice device, int rssi, byte[] bytes)）

该接口的返回值说明如下：

```
1. private BluetoothAdapter.LeScanCallback leScanCallback = new BluetoothAdapter.LeScanCallback() {
2.
3.     @Override
4.     public void onLeScan(BluetoothDevice device, int rssi, byte[] bytes) {
5.         //处理搜索到的设备
6.     }
7. };
```

注意事项：1.保证蓝牙设备有电

2. 如要筛选蓝牙设备（厂商 ID == 0xFF01），具体请参考四、其他，筛选相关

### 3. demo 中包含筛选长连接的示例

#### 3.1.2 停止搜索

接口功能：关闭蓝牙搜索功能。

接口声明：

```
1. BLEUtils.stopLeScan(Context context, BluetoothAdapter.LeScanCallback leScanCallback);
```

注意事项：调用此接口，需保证与戒指处于连接状态

参数说明：context：上下文      leScanCallback：蓝牙搜索的回调

返回值：无

#### 3.1.3 连接设备

接口功能：发起连接蓝牙设备。

接口声明：

```
1. BLEUtils.connectLockByBLE(Context context, BluetoothDevice bluetoothDevice);
```

注意事项：1.调用此接口，需保证与戒指处于连接状态

2.扫描连接时会自动判断是否为长连接，在已经长连接下去重连时，需手动输入 bool 值确认是否重连（demo 里有示例）

参数说明：context：上下文      bluetoothDevice：蓝牙设备

返回值：

```
1. @Override
2. public void lmBleConnecting(int code) {
3.     //正在连接
4.
5. }
6.
7. @Override
8. public void lmBleConnectionSucceeded(int code) {
9.     //连接成功
10. }
11.
12. @Override
13. public void lmBleConnectionFailed(int code) {
```

```
14. //连接失败
15. }
```

### 3.1.4 断开蓝牙

接口功能：断开设备。

接口声明：

```
1. BLEUtils.disconnectBLE(Context context);
```

注意事项：调用此接口，需保证与戒指处于连接状态

参数说明：context：上下文

返回值：无

## 3.2 指令功能（LmAPI）

此类是使用戒指功能的公共类，戒指的功能通过该类直接调用即可,数据反馈除了测量和历史记录外 统一 由 IResponseListener 接口反馈

### 3.2.1 同步时间

接口功能：同步时间，调用此接口，会获取手机当前时间同步给戒指，保持时间同步。

接口声明：

```
1. LmAPI.SYNC_TIME();
```

注意事项：同步时间和读取时间共用一个返回值。调用此接口，需保证与戒指处于连接状态

参数说明：无

返回值（void syncTime(byte datum,byte[] time)）

| 参数名称  | 类型     | 示例值   | 说明                   |
|-------|--------|-------|----------------------|
| datum | byte   | 0 或 1 | 0 代表同步成功<br>1 代表读取时间 |
| time  | byte[] | null  | 同步时间不会返回 byte[]      |

### 3.2.2 读取时间

接口功能：读取时间，调用此接口，会获取戒指当前时间。

接口声明：

注意事项：同步时间和读取时间共用一个返回值。调用此接口，需保证与戒指处于连接状态

参数说明：无

```
1. LmAPI.READ_TIME();
```

返回值（void syncTime(byte datum, byte[] time)）

| 参数名称  | 类型     | 示例值  | 说明                   |
|-------|--------|--|----------------------|
| datum | byte   | 0 或 1  | 0 代表同步成功<br>1 代表读取时间 |
| time  | byte[] | [48, -23, -1, 83, -111, 1, 0, 0, 8] =<br>1723691166000 | 读取时间成功，需转化为时间戳       |

### 3.2.3 版本信息

接口功能：版本信息，获取戒指的版本信息。

接口声明：

```
1. LmAPI.GET_VERSION((byte) 0x00); //0x00 获取软件版本, 0x01 获取硬件版本
```

注意事项：调用此接口，需保证与戒指处于连接状态

参数说明：type: 0x00 获取软件版本，0x01 获取硬件版本

返回值（void VERSION(byte type, String version)）

| 参数名称    | 类型     | 示例值     | 说明   |
|---------|--------|---------|--|
| type    | byte   | 0 或 1   | <ul style="list-style-type: none"> <li>0 代表软件</li> <li>1 代表硬件</li> </ul> |
| version | String | 1.0.0.1 | <ul style="list-style-type: none"> <li>版本号</li> </ul>                    |

### 3.2.4 电池电量

接口功能：获取电池电量、 电池状态。

接口声明：

```
1. LmAPI.GET_BATTERY((byte) 0x00); //0x00 获取电量, 0x01 获取充电状态
```

注意事项：调用此接口，需保证与戒指处于连接状态

参数说明：type: 0x00 获取电量，0x01 获取充电状态

返回值（void battery(byte status, byte datum)）

| 参数名称   | 类型   | 示例值   | 说明   |
|--------|------|-------|--|
| status | byte | 0 或 1 | <ul style="list-style-type: none"> <li>● 0 代表电池电量</li> <li>● 1 代表充电状态</li> </ul> |
| datum  | byte | 1     | <ul style="list-style-type: none"> <li>● 电量</li> </ul>                           |

### 3.2.5 读取步数

接口功能：获取当天累计步数。

接口声明：

```
1. LmAPI.STEP_COUNTING ( )
```

注意事项：调用此接口，需保证与戒指处于连接状态

参数说明：无

返回值（void stepCount(byte[] bytes, byte subCmd)）

| 参数名称   | 类型     | 示例值  | 说明   |
|--------|--------|------|--|
| bytes  | byte[] | 3303 | <ul style="list-style-type: none"> <li>● 步数 819</li> </ul>                       |
| subCmd | byte   | 0, 1 | <ul style="list-style-type: none"> <li>● 0 代表步数</li> <li>● 1 代表清除步数成功</li> </ul> |

### 3.2.6 清除步数

接口功能：清除步数。

接口声明：

```
1. LmAPI.CLEAR_COUNTING ()
```

注意事项：调用此接口，需保证与戒指处于连接状态

参数说明：无

返回值：参考上条

### 3.2.7 恢复出厂设置

接口功能：恢复出厂设置

接口声明：

```
1. LmAPI.RESET ()
```

注意事项：调用此接口，需保证与戒指处于连接状态

参数说明：无

返回值：无，有回调 reset 方法即认为成功

### 3.2.8 采集周期设置

接口功能：采集周期设置

接口声明：

```
1. LmAPI.SET_COLLECTION (collection) //采集周期，单位秒
```

注意事项：调用此接口，需保证与戒指处于连接状态

参数说明：collection：采集间隔，单位秒

返回值：(void collection(byte[] bytes, byte subCmd))

| 参数名称  | 类型     | 示例值      | 说明    |
|-------|--------|----------|-------|
| bytes | byte[] | b0040000 | 1200秒 |



|        |      |      |                    |
|--------|------|------|--------------------|
| subCmd | byte | 0或者1 | 0代表设置成功<br>1代表设置失败 |
|--------|------|------|--------------------|

### 3.2.9 采集周期读取

接口功能：采集周期读取

接口声明：

```
1. LmAPI.GET_COLLECTION () //采集周期，单位秒
```

注意事项：调用此接口，需保证与戒指处于连接状态

参数说明：无

返回值：(void collection(byte[] bytes, byte subCmd))

| 参数名称   | 类型     | 示例值      | 说明                    |
|--------|--------|----------|-----------------------|
| bytes  | byte[] | b0040000 | 采集时间间隔，单位秒<br>如：1200s |
| subCmd | byte   | 0或者1     | 0代表设置成功<br>1代表设置失败    |

### 3.2.10 测量心率

接口功能：测量心率。

接口声明：

```
1. LmAPI.GET_HEART_ROTA (byte waveForm, byte  
acqTime, IHeartListener iHeartListener)
```

注意事项：调用此接口，需保证与戒指处于连接状态

参数说明：

waveForm：是否配置波形 0 不上传 1 上传

acqTime：采集时间（byte）30 是正常时间, 0 为一直采集

iHeartListener：此接口是测量数据的监听

返回值:

```
1.  LmAPI.GET_HEART_ROTATE((byte) 0x01, (byte)0x30, new IHeartListener() {
2.      @Override
3.      public void progress(int progress) {
4.          setMessage("正在测量心率..." + String.format
5. ("%" + 0 + 2 + "d%" + 0 + 0, progress));
6.      }
7.
8.      @Override
9.      public void resultData(int heart, int heartRota, int yaLi, int temp)
10.     {
11.         if (colorFragment != null) {
12.             colorFragment.heartAndRota(heart, heartRota, yaLi, temp);
13.         }
14.     }
15.     @Override
16.     public void waveformData(byte seq, byte number, String waveData) {
17.         //心率返回波形图数据分析: waveData
18.     }
19.     @Override
20.     public void error(int value) {
21.         switch (value) {
22.             case 0:
23.                 dismissProgressDialog();
24.                 ToastUtils.show("未佩戴");
25.                 break;
26.             case 2:
27.                 dismissProgressDialog();
28.                 ToastUtils.show("充电中不允许采集");
29.                 break;
30.             case 4:
31.                 dismissProgressDialog();
32.                 ToastUtils.show("繁忙, 不执行");
33.                 break;
34.             default:
35.                 break;
36.         }
37.     }
38.
39.     @Override
40.     public void success() {
41.         dismissProgressDialog();
42.     }
```

```
43. });  
44.
```

### 3.2.11 测量血氧

接口功能：测量血氧。

接口声明：

```
1. LmAPI.GET_HEART_Q2 (byte waveForm, IQ2Listener iQ2Listener)
```

注意事项：调用此接口，需保证与戒指处于连接状态

参数说明：

**waveForm**：是否配置波形 0 不上传 1 上传

**IQ2Listener**：此接口是测量数据的监听

返回值：

```
1. LmAPI.GET_HEART_Q2(new IQ2Listener() {  
2.     @Override  
3.     public void progress(int progress) {  
4.         setMessage("正在测量血氧..." + String.format("%02d%", progress));  
5.     }  
6.  
7.     @Override  
8.     public void resultData(int heart, int q2, int temp) {  
9.         if (colorFragment != null) {  
10.            colorFragment.updateData(heart, q2, temp);  
11.        }  
12.    }  
13.  
14.    @Override  
15.    public void waveformData (byte seq, byte number, String waveData) {  
16.        //血氧返回波形图数据分析: waveData  
17.    }  
18.    @Override  
19.    public void error(int value) {  
20.        switch (value) {  
21.            case 0:  
22.                dismissProgressDialog();  
23.                ToastUtils.show("未佩戴");  
24.                break;  
25.            case 2:  
26.                dismissProgressDialog();
```

```
27.         ToastUtils.show("充电中不允许采集");
28.         break;
29.     case 4:
30.         dismissProgressDialog();
31.         ToastUtils.show("繁忙，不执行");
32.         break;
33.     default:
34.         break;
35.     }
36. }
37.
38. @Override
39. public void success() {
40.     dismissProgressDialog();
41. }
42. });
```

### 3.2.12 测量温度

接口功能： 测量温度。

接口声明：

```
1. LmAPI.GET_HEART_Q2 (IQ2Listener iq2Listener)
```

注意事项：调用此接口 ， 需保证与戒指处于连接状态

参数说明：IQ2Listener: 此接口是测量数据的监听

返回值：同上。测量血氧时同时会返回温度

### 3.2.13 历史记录管理

接口功能：读取历史记录。

接口声明：

```
1. LmAPI.READ_HISTORY (int type,IHistoryListener iHistoryListener)
```

注意事项：调用此接口 ， 需保证与戒指处于连接状态

参数说明：type: 1,获取全部历史记录；0， 获取未上传的历史记录

返回值：

```
1. LmAPI.READ_HISTORY(type, new IHistoryListener() {
2.     @Override
3.     public void error(int code) {
```

```
4.         handler.removeMessages(0x99);
5.         dismissProgressDialog();
6.         switch (code) {
7.             case 0:
8.                 ToastUtils.show("正在测量中,请稍后重试");
9.                 break;
10.            case 1:
11.                ToastUtils.show("正在上传历史记录,请稍后重试");
12.                break;
13.            case 2:
14.                ToastUtils.show("正在删除历史记录,请稍后重试");
15.                break;
16.            default:
17.                break;
18.        }
19.    }
20.
21.    @Override
22.    public void success() {
23.        //同步完成
24.    }
25.
26.    @Override
27.    public void progress(double progress, com.lm.sdk.mode.HistoryDataBean
        dataBean) {
28.        //处理历史数据
29.    }
30. });
```

### 3.2.14 清空历史数据

接口功能：清空历史数据。

接口声明：

```
1.  LmAPI.CLEAN_HISTORY ()
```

注意事项：调用此接口，需保证与戒指处于连接状态

参数说明：无

返回值：无

### 3.2.15 血压测试算法

接口功能：清空历史数据。

接口声明：

```
1. LmAPI.GET_BPwaveData()
```

注意事项：**戒指固件必须支持，否则无法使用。**调用此接口，需保证与戒指处于连接状态

参数说明：无

返回值(byte seq,byte number,String waveDate)

| 参数名称     | 类型     | 示例值   | 说明         |
|----------|--------|---|------------|
| seq      | byte   | 0   | ● 序号 0     |
| number   | byte   | 10  | ● 有 10 个数据 |
| waveDate | String | green/绿光:14289393<br>ir/红外:10108995<br>cur_green/绿光电<br>流:4704 cur_ir/红外电<br>流:4704 | ● 无        |

### 3.2.16 实时 PPG 血压测量

接口功能：实时测量血压值和 500hz 的原始波形

接口声明：

```
1. LmAPI.GET_REAL_TIME_BP (byte time,byte isWave,byte  
isProgress,IRealTimePPGBpListener iRealTimePPGBpListener)
```

注意事项：**戒指固件必须支持，否则无法使用。**调用此接口，需保证与戒指处于连接状态

参数说明：

time：采集时间,byte 类型，默认 30s

isWave:是否上传波形。0：不上传，1：上传

isProgress：是否上传进度。0：不上传，1：上传

```

1.  LmAPI.GET_REAL_TIME_BP((byte) 0x30, (byte) 1, (byte) 1, new IRealTimePPGBp
    Listener() {
2.
3.      @Override
4.      public void progress(int progress) {
5.          //进度
6.      }
7.
8.      @Override
9.      public void bpResult(byte type) {
10.         //[0]:舒张压
11.         //[1]:收缩压
12.      }
13.
14.      @Override
15.      public void resultData(String bpData) {
16.          //bpData 包含红外值
17.      }
18.  });

```

### 3.2.17 实时 PPG 血压停止采集

接口功能：停止采集

接口声明：

```
1.  STOP_REAL_TIME_BP()
```

参数说明：无

回调：

```

1.  @Override
2.  public void stopRealTimeBP(byte isSend) {
3.      if(isSend == (byte)0x01){
4.          Logger.show("TAG", "停止采集已发送");
5.      }
6.  }

```

### 3.2.18 设置蓝牙名称

接口功能：设置蓝牙名称

接口声明：

```
1.  Set_BlueTooth_Name(String name)
```

参数说明：

Name:蓝牙名称，不超过 12 个字节，可以为中文、英文、数字，即 4 个汉字或者 12 个英文

注：设置蓝牙名称后，广播不会立即改变，需要等待一段时间

回调：

```

1.  @Override
2.  public void setBlueToolName(byte data) {

```

```
2.         if(data == (byte)0x00){
3.             Logger.show("TAG", "设置失败");
4.         }else if(data == (byte)0x01){
5.             Logger.show("TAG", "设置成功");
6.         }
7.     }
```

### 3.2.19 获取蓝牙名称

接口功能：设置蓝牙名称

接口声明：

```
1. Get_BlueTooth_Name()
```

参数说明：无

回调：

```
1. @Override
2.     public void readBlueToolName(byte len, String name) {
3.         Logger.show("TAG", "蓝牙名称长度: " + len + " 蓝牙名称: " + name);
4.     }
```

### 3.2.20 心率测量停止

接口功能：停止正在测量的心率

接口声明：

```
1. LmAPI.STOP_HEART()
```

参数说明：无

回调：

```
1. @Override
2.     public void stopHeart(byte data) {
3.         Logger.show("TAG", "stop success");
4.     }
```

### 3.2.21 血氧测量停止

接口功能：停止正在测量的血氧

接口声明：

```
1. LmAPI.STOP_Q2()
```

参数说明：无

回调：

```
1. @Override
2.     public void stopQ2(byte data) {
3.         Logger.show("TAG", "stop success");
4.     }
```



### 3.2.22 一键获取状态

接口功能：一键获取系统支持的功能，简化版的接口集合，会返回电量、固件版本、采集周期等  
接口声明：

```
1. LmAPI.SYSTEM_CONTROL()
```

参数说明：无

回调：

```
1. @Override
2.     public void SystemControl(SystemControlBean systemControlBean) {
3.         postView("\nSystemControl: "+systemControlBean.toString());
4.     }
```

### 3.2.23 语音录制

接口功能：

接口声明：

```
1. LmAPI.SET_AUDIO(byte data)
```

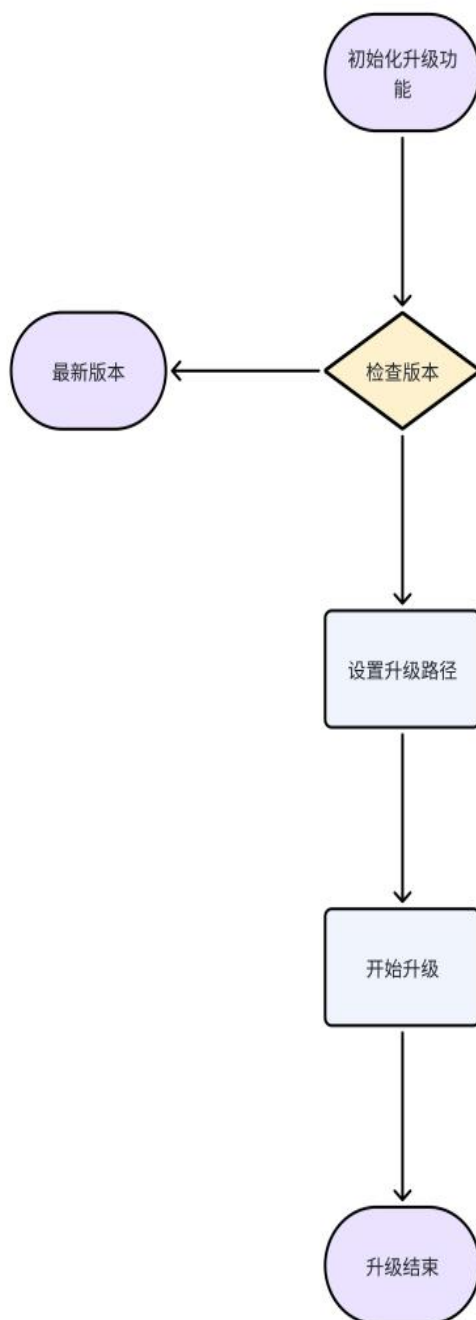
参数说明：0 代表关闭，1 代表打开

回调：

```
1. @Override
2.     public void CONTROL_AUDIO(short[] bytes) {
3.         postView("\n 音频结果: " + Arrays.toString(bytes));
4.     }
```

注：返回的数据是有符号短整型

### 3.3 固件升级（OTA）



#### 3.3.1 检查版本

接口功能：检查固件版本是否是最新。

接口声明：

```
1. OtaApi.checkVersion(String version, VersionCallback versionCallback);
```

注意事项：调用此接口，需保证与戒指处于连接状态

参数说明：version：当前戒指的版本号

versionCallback：最新版本信息回调

返回值：

```
1. OtaApi.checkVersion(version, new VersionCallback() {
2.     @Override
3.     public void success(String newVersion) {
4.         //newVersion: 云端最新版本号
5.         if (!StringUtils.isEmpty(newVersion)){
6.             //有新版本
7.         }else{
8.             //已是最新版本
9.         }
10.    }
11.
12.    @Override
13.    public void error() {
14.        //获取最新版本失败
15.    }
16.});
```

### 3.3.2 开始升级

接口功能：执行固件升级。

接口声明：

```
1. void startUpdate(BluetoothDevice bluetoothDevice, int rssi, LmOTACallback
   otaCallback)
```

注意事项：调用此接口，需保证与戒指处于连接状态

参数说明：bluetoothDevice：当前要升级的设备

rssi：设备信号值

otaCallback：升级回调

返回值：

```
1. OtaApi.startUpdate(App.getInstance().getDeviceBean().getDevice(), App.getI
   nstance().getDeviceBean().getRssi(), new LmOTACallback() {
2.     @Override
3.     public void onDeviceStateChange(int i) {
```

```
4.         //设备状态回调
5.     }
6.
7.     @Override
8.     public void onProgress(int i, int i1) {
9.         //升级进度
10.    }
11.
12.    @Override
13.    public void onComplete() {
14.        //升级完成
15.    }
16. });
```

## 3.4 数据库相关 (DataApi)

### 3.4.1 查询历史记录

接口功能：查询指定时间的历史数据

接口声明：

```
1. //查询历史数据
2. List<HistoryDataBean> queryHistoryData(long dayBeginTime,long dayEndTime,String mac)
3. //查询历史数据按照时间进行正序
4. List<HistoryDataBean> queryHistoryDataOrderByTimeAsc(long dayBeginTime,long dayEndTime,String mac)
5. //查询历史数据按照步数进行倒叙
6. List<HistoryDataBean> queryHistoryDataOrderByStepCountDesc(long dayBeginTime,long dayEndTime,String mac)
```

注意事项：需开启相应权限

参数说明：dayBeginTime ：开始时间戳，单位：秒

dayEndTime ：结束时间戳，单位：秒

mac ：设备的 MAC 地址

返回值：

```
1. public class HistoryDataBean{
2.
3.     @Id
4.     private Long id;
```

```
5.
6.     private String mac;
7.         //总数据包数 4 个字节
8.     private long totalNumber;
9.         // 当前第几包 4 个字节
10.    private long indexNumber;
11.        // 当前记录时间 4 个字节
12.    private long time;
13.        //今天累计步数 2 个字节
14.    private int stepCount;
15.        // 心率 1 个字节
16.    private int heartRate;
17.        // 血氧 1 个字节
18.    private int bloodOxygen;
19.        //心率变异性 1 个字节
20.    private int heartRateVariability;
21.        //精神压力指数 1 个字节
22.    private int stressIndex;
23.        //温度 2 个字节
24.    private int temperature;
25.        //运动激烈程度 1 个字节
26.    private int exerciseIntensity;
27.    /**
28.     * 睡眠类型 1 个字节
29.     * 0: 无效
30.     * 1: 清醒
31.     * 2: 浅睡
32.     * 3: 深睡
33.     * 4: 眼动期
34.     */
35.    private int sleepType;
36.        // 保留 2 个字节
37.    private int reserve;
38.        //RR 间期 1 个字节
39.    private int rrCount;
40.        //RR 数组数据 1 个字节
41.    private byte[] rrBytes;
42. }
```

### 3.4.2 清空历史数据

接口功能：清空全部历史数据。

接口声明：

```
1. void deleteHistoryData();
```

注意事项：需开启相应权限

参数说明：无

返回值：无

## 3.5 逻辑算法相关 (LogicalApi)

### 3.5.1 计算距离、卡路里

接口功能：根据步数计算距离、卡路里。

接口声明：

```
1. DistanceCaloriesBean calculateDistance(int stepCount, double height, double weight);
```

注意事项：需开启相应权限

参数说明：stepCount：步数

height：身高 单位 cm

weight：体重 单位 kg

返回值：

```
1. public class DistanceCaloriesBean {  
2.     //距离，单位米  
3.     private double distance;  
4.     //卡路里，单位卡  
5.     private double kcal;  
6. }
```

### 3.5.2 计算睡眠数据

接口功能：计算指定时间睡眠数据。

接口声明：

```
1. SleepBean calculateSleep(String date, String mac, int type);
```

注意事项：需开启相应权限

参数说明：date：日期，格式为 YYYY-MM-DD HH:mm:ss

mac : 设备蓝牙 MAC

type: 暂时无效

返回值:

```
1. public class SleepBean{
2.     // 零星睡眠小时
3.     int hours = 0;
4.     // 零星睡眠分钟
5.     int minutes = 0;
6.     // 全部睡眠小时
7.     int allHours = 0;
8.     // 全部睡眠分钟
9.     int allMinutes = 0;
10.    // 全部睡眠小时
11.    int sleepHours = 0;
12.    // 全部睡眠分钟
13.    int sleepMinutes = 0;
14.    // 深度睡眠时间
15.    long highTime = 0;
16.    // 浅度睡眠时间
17.    long lowTime = 0;
18.    // 眼动时间
19.    long ydTime = 0;
20.    // 清醒时间
21.    long qxTime = 0;
22.    // 入睡时间戳
23.    long startTime = 0;
24.    // 清醒时间戳
25.    long endTime = 0;
26.    // 睡眠数据
27.    private List<HistoryDataBean> historyDataBeanList;
28. }
```

数据库中数据

```
1. public class HistoryDataBean{
2.
3.     @Id
4.     private Long id;
5.     private String mac;
6.     // 总数据包数 4 个字节
7.     private long totalNumber;
8.     // 当前第几包 4 个字节
9.     private long indexNumber;
10.    // 当前记录时间 4 个字节
```

```
11.     private long time;
12.         // 今天累计步数 2 个字节
13.     private int stepCount;
14.         // 心率 1 个字节
15.     private int heartRate;
16.         // 血氧 1 个字节
17.     private int bloodOxygen;
18.         // 心率变异性 1 个字节
19.     private int heartRateVariability;
20.         // 精神压力指数 1 个字节
21.     private int stressIndex;
22.         // 温度 2 个字节
23.     private int temperature;
24.         // 运动激烈程度 1 个字节
25.     private int exerciseIntensity;
26.     /**
27.      * 睡眠类型 1 个字节
28.      * 0: 无效
29.      * 1: 清醒
30.      * 2: 浅睡
31.      * 3: 深睡
32.      * 4: 眼动期
33.      */
34.     private int sleepType;
35.         // 保留 2 个字节
36.     private int reserve;
37.         // RR 间期 1 个字节
38.     private int rrCount;
39.         // RR 数组数据 1 个字节
40.     private byte[] rrBytes;
41. }
```

## 四、 其他

注：使用戒指 API 前，应先查看戒指状态

### 1、 筛选相关

血氧戒指设备以 XXXXXX 的名字进行广播。XXX 为任何字符，广播间隔为 500ms。本例中 XXX 为 BCL603。

广播包的长度必须是 31 个字节，如果不到 31 个字节，则剩下的全用 0 填充 补全，这部分的数据是无效的

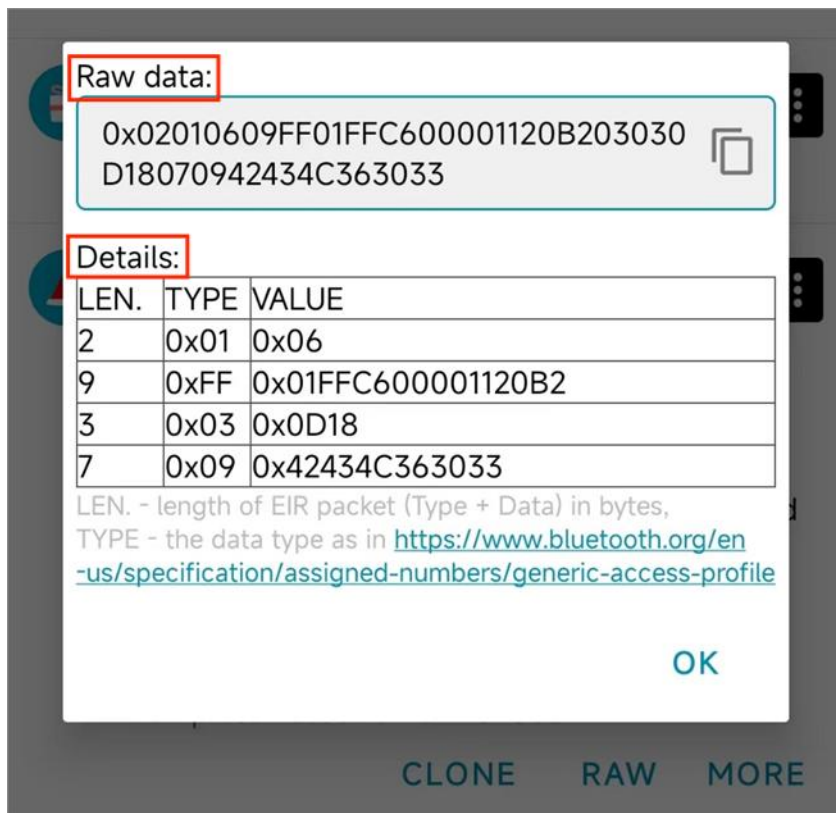


## 1.1 广播数据单元

广播包中包含若干个广播数据单元，广播数据单元也称为 AD Structure。

**广播数据单元 = 长度值 Length + AD type + AD Data**

长度值 Length 只占一个字节，并且位于广播数据单元的**第一个字节**。



### 1.1.1 Raw data:

0x 代表这串字符串是十六进制的字符串。两位十六进制数代表一个字节。因为两个字符组成的十六进制字符串最大为 FF，即 255，而 Java 中 byte 类型的取值范围是 -128 到 127，刚好可以表示一个 255 的大小。所以两个十六进制的字符串表示一个字节。

继续查看报文内容，开始读取第一个广播数据单元。读取第一个字节:0x02,转换为十进制就是 2，即表示后面的 2 个字节是这个广播数据单元的数据内容。超过这 2 个字节的数据内容后，表示是一个新的广播数据单元。

而第二个广播数据单元，第一个字节的值是 0x09,转换为十进制就是 9，表示后面 9 个字节为第二个广播数据单元。

而第三个广播数据单元，第一个字节的值是 0x03,转换为十进制就是 3，表示后面 3 个字节为第三个广播数据单元。

以此类推。

在广播数据单元的数据部分中，第一个字节代表数据类型（AD type），决定数据部分表示的是什么数据。（即广播数据单元第二个字节为 AD type）



- (1)、Type = 0x01 表示设备 LE 物理连接。
- (2)、Type = 0xFF 表示厂商数据。前两个字节表示厂商 ID,即厂商 ID 为 0xFF01。后面的为厂商数据,具体由用户自行定义
- (3)、Type = 0x03 表示完整的 16bit UUID。其值为 0x0D18。
- (4)、Type = 0x09 表示设备的全名,例如: 0x42434C363033 转 byte[]再转字符串即为“BCL603”

注：数据传输方式为小端模式

[illegible]

已知该数据的格式和含义，根据规则设置筛选条件为厂商 ID == “FF01” 即可  
或直接对接收到的数据验证 “01FF”

新增广播 UUID[1]，“0018”，在广播 FF01 之后  
新增点在于“0018”，二进制表示 0000 0000 0001 1000（小端模式），赋予更多含义  
释意：

0:不支持一键获取状态指令的版本。

1:支持一键获取状态指令的版本。

## 2、 可能会遇到的问题

资料中带有简单 demo，可以先查看简单 demo 使用 SDK 的逻辑，再进行自己的开发

### 2.1 版本相关

Gradle 版本。可在 gradle-wrapper.properties 里修改

```
1. distributionBase=GRADLE_USER_HOME
2. distributionPath=wrapper/dists
3. distributionUrl=https\://services.gradle.org/distributions/gradle-8.0-all.zip
4. zipStoreBase=GRADLE_USER_HOME
5. zipStorePath=wrapper/dists
```

可在以下网址下载所需版本，将 zip 放在 `wrapper/dists` 对应的路径下（放在随机码文件夹下，记得清空原有内容），重新 sync 一下

[Gradle Distributions](#)

### 2.2 Gradle 4.0 以上导致 Xpopup 无法使用

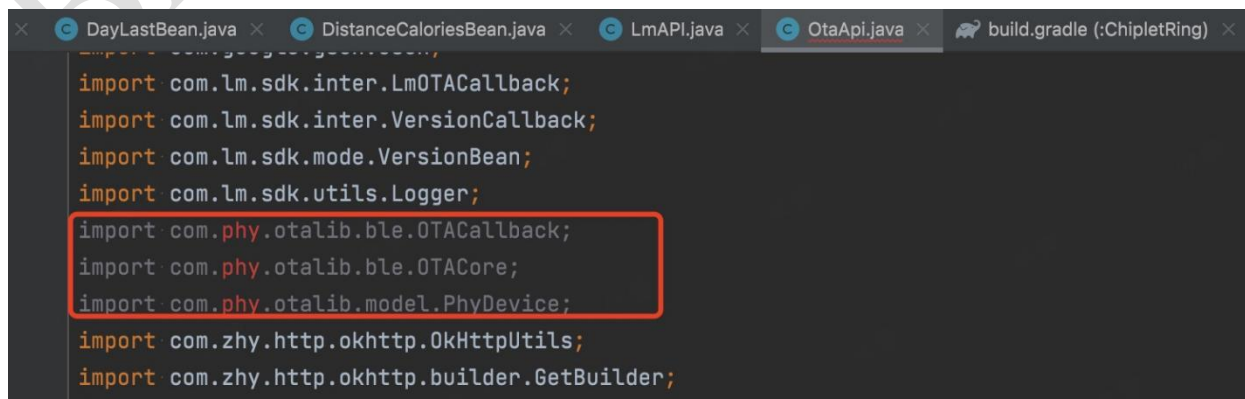
Xpopup 是个第三方弹窗框架，换为普通弹窗可以解决问题

### 2.3 不需要界面

ringSDK1.0.2 已支持不需要界面，在 service 做扫描、连接等功能

### 2.4 OTA 类引用未找到

可能会出现以下情况：



官方的其他依赖库，若找不到，可联系勇芯开发人员，要原始类原始 jar 包已放到 SDK 的 OTA 类文件夹下

### 3、 硬件算法逻辑或固件相关

#### 3.1 戒指相关

问：戒指多久存一次数据

答：5 分钟

问：OTA 升级会清除数据吗

答：会

问：戒指里的数据可以存几天

答：7 天，7 天后自动覆盖

问：恢复出厂设置是只恢复戒指吗

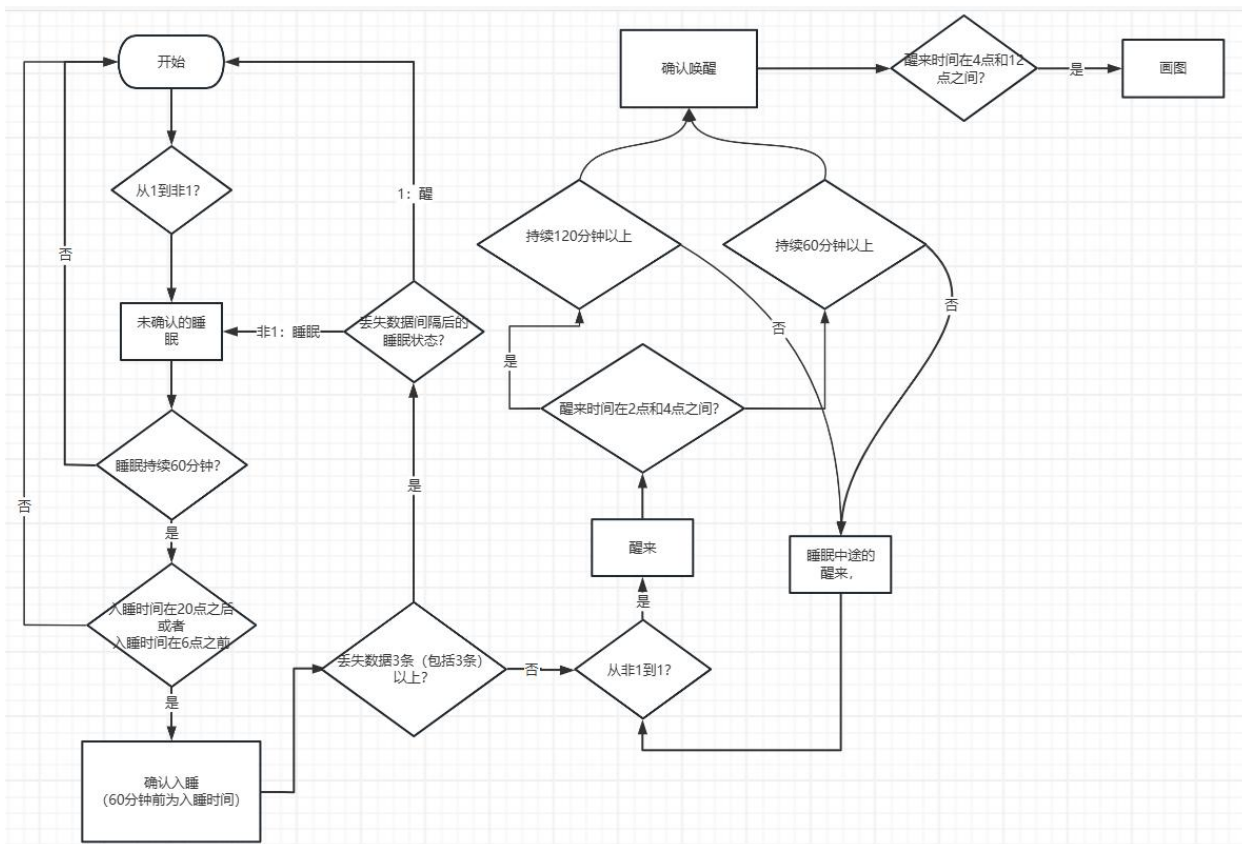
答：对，只针对戒指硬件进行恢复

#### 3.2 算法相关

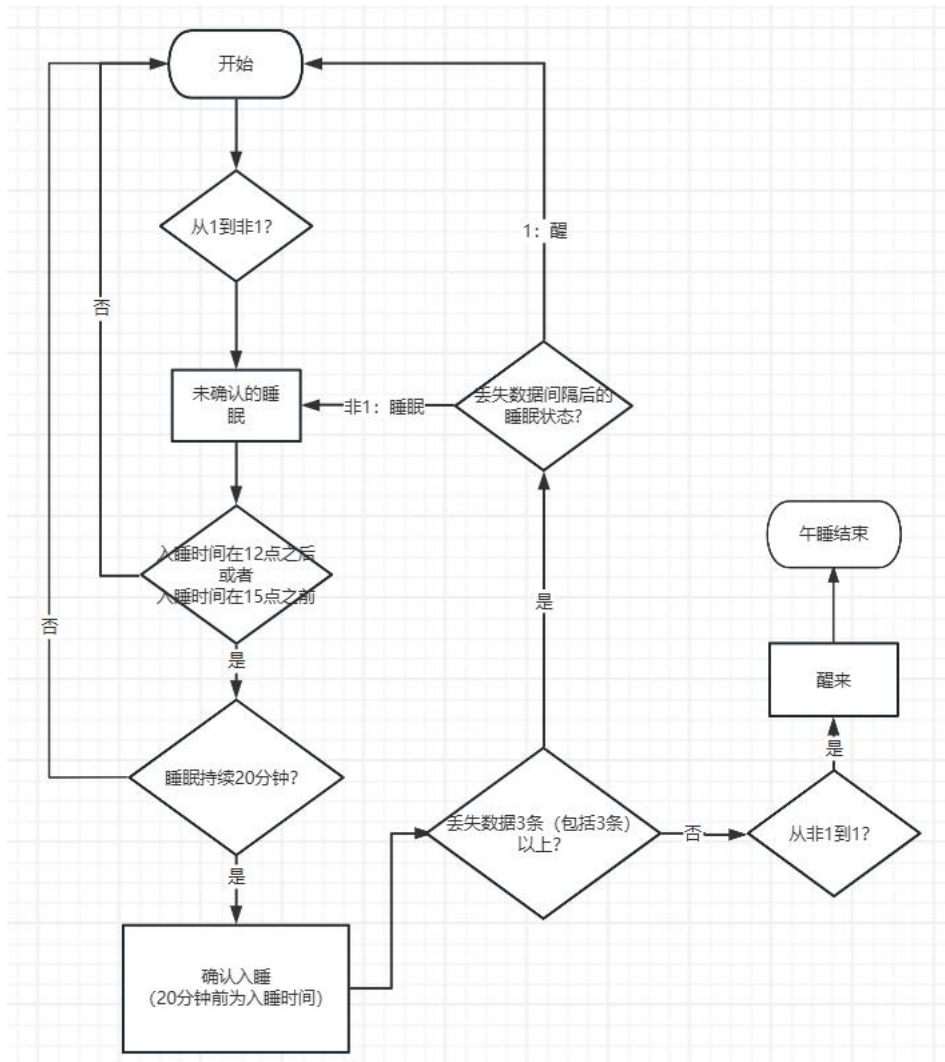
问：为什么使用 SDK 的心率/血氧测量时总显示超时

答：戒指充电时，无法进行心率血氧测量

### 3.3 睡眠逻辑图



### 3.4 午睡逻辑图



## 4、 Q&A

Q: 读历史记录过程中，是上报一条标记一条已同步，还是全部上传了整体标记？

A: 一条一条

Q: 采集周期设置有什么限制

A: 采集周期单位为秒，正常值最小为 60s，为 0 时代表关闭采集。