

# 美的连医疗器械 Android 端 SDK

## 1.0.3 版本

-----1.0.0 版本

-----1.0.1 移除了 ProcotolService，添加了读心包的协议

-----1.0.2 修改了红外体温计测量值差 0.1 的问题

-----1.0.3 新增了血氧仪测量呼吸的代码，以及血氧接口分离

-----1.0.4 修改红外体温计测量高温显示不一致问题

该 SDK 帮助您快速实现 MEDXING 蓝牙设备或串口设备数据的解析，目前支持的设备有 6 种

- 1.MEDXING-ECG 心电仪
- 2.MEDXING-Spo2 血氧仪
- 3.MEDXING-IRT 红外体温计
- 4.MEDXING-BGM 血糖仪
- 5.MEDXING-NIBP 血压计
- 6.MEDXING-BMI 体脂称
- 7.MEDXING-ECG 心电仪(读心宝)

## AndroidStudio 项目中导入 Android 端 SDK 的步骤

第一步：

在 AndroidStudio 的项目中 app 目录下创建 aars 的文件夹，将名称为 MedxingDev.aar 的文件，放入该文件夹中

第二步：

在 app 目录底下的 build.gradle 底下的 android{} 内加入以下代码

```
allprojects{
    repositories{
        jcenter()
    }
}
```

```

        flatDir{
            dir'aars'
        }
    }
}

```

在 app 目录底下的 build.gradle 底下的 dependencies{} 内加入以下代码

```

//AndroidStudio3.0 之后
implementation (name:'medxingble-debug',ext:'aar')
//AndroidStudio3.0 之前
//compile (name:'medxingble-debug',ext:'aar')

```

第三步:

Sync Now 后, 即可调用 SDK 中的方法了

## SDK 的使用方法

第一步:

注册解析类 Service, 在 AndroidManifest.xml 中加入

```

<service android:name="sdk.nurse.medxing.com.medxingble.resolve.ProtocolService" />

```

注:1.0.1 把这个服务写成了内部类,并删除了这个服务的代码,不需要注册该服务

第二步:

在蓝牙或者串口接收数据的地方, 获取 SDK 中的管理类, 并实现 `setOnResolveListener<T>` 泛型的公共接口来实现监听不同设备数据的解析结果,<T>可以传入需要解析的实体类

```

private ResolveManager resolveManager;
protected void onCreate(@Nullable Bundle savedInstanceState) {
    resolveManager = ResolveManager.getInstance(this);
    resolveManager.setOnResolveListener(new
ResolveManager.OnResolveListener<BmiResolve>(){
    @Override
    public void onResolve(BmiResolve bmiResolve) {
        bmiBodyInfo.loadData(bmiResolve,176,22,1);
        id_text_bmi.setText(bmiResolve.toString() + "--" +bmiBodyInfo.toString());
    }
});
}

```

接口中的 onResolve 就是我们解析出来的实体类, 里面有对应的测量值, 直接填充到 UI 上既可以

下面是统一使用公共的泛型接口<T>可以传入需要解析的实体类

```
/**
 * 泛型的公共接口
 */
public interface OnResolveListener<T>{
    void onResolve(T t);
}
```

第三步:

在接收到蓝牙或者串口数据后, 通过 `pushData` 的方法, 传入管理类中

```
byte[] bytes =
intent.getByteArrayExtra(BluetoothService.BLUETOOTH_DEVICES_NOTIFY_DATA);
resolveManager.pushData(bytes);
```

第四步:

程序员的美德, 使用完之后, 记得 `close()` 节省开支

```
resolveManager.close();
```

## 设备对应的 Resolve 类讲解

### 1.BgmResolve 血糖解析类

`String type`, 是测量进行到了哪一步, 他的值是 `BgmResolve` 中的常量

```
public static final String BGM_TYPE_NOT_CONNECT = "未连接";
public static final String BGM_TYPE_READY = "设备就绪";
public static final String BGM_TYPE_PAPER_READY = "请插入试纸";
public static final String BGM_TYPE_PAPER_READYING = "检查试纸中...";
public static final String BGM_TYPE_PAPER_ERROR = "试纸无效, 请换张试纸后重试";
public static final String BGM_TYPE_PAPER_READY_OK = "请采血";
public static final String BGM_TYPE_MEASURE = "测量中...";
public static final String BGM_TYPE_MEASURED = "测量完成";
public static final String BGM_TYPE_MEASURE_FAILURE = "测量失败";
```

根据以上不同的步骤, UI 中可以提示用户进行相关的操作及指示

`String countDown`, 是测量中...阶段的倒计时, 时长为 5 秒

`float mmol`, 是测量完成后返回的血糖浓度

`getLevelStr(int measurerStatus,float mmol)`, 该方法判断测量者的血糖, 是否处于正常范围, 第一个参数为下面常量, 第二个参数为血糖浓度

```
//测量者状态
public static final int BGM_MEASURER_BEFORE_30 = 1;//餐前三十分钟
public static final int BGM_MEASURER_BEFORE_60 = 2;//餐前一小时
public static final int BGM_MEASURER_BEFORE_120 = 3;//餐前两小时
public static final int BGM_MEASURER_BEFORE_180 = 4;//餐前三小时
```

## 2.BmiResolve 体脂秤

String type，是测量进行到了哪一步，他的值是 WeighResolve 中的常量

```
public static final String WEIGH_STATUS_UNCONNECTED = "未连接";
public static final String WEIGH_STATUS_READY = "设备就绪";
public static final String WEIGH_STATUS_MEASURE = "测量中...";
public static final String WEIGH_STATUS_MEASURED = "测量完成";
public static final String BWEIGH_STATUS_MEASURE_FAILURE = "测量失败";
```

float weight，是测量完成后传回来的体重

int impedance，是测量完成后传回来的阻抗值，用来计算身体的成分主要参数

### 2.1 WeighInfo 身体成分解析

calculate(WeighResolve weighResolve,float height,int age,int sex)，该方法内含专业的身体成分分析算法，传入 weighResolve，与身高，年龄，性别（0 女，1 男），可以得出以下身体成分信息

```
private float bodyfat;//脂肪
private float bodywater;//体水分
private float muscle;//肌肉
private float bone;//骨头
private float bmi;//身体指数
private float visceralfat;//内脏脂肪
private float waistcur;//腰围
private int dci;//dci
```

## 3.EcgResolve 心电解析

String type，是测量进行到了哪一步，他的值是 EcgResolve 中的常量

```
public static final String ECG_TYPE_NOT_CONNECT = "未连接";
public static final String ECG_TYPE_READY = "设备就绪";
public static final String ECG_TYPE_MEASURE = "测量中...";
public static final String ECG_TYPE_MEASURED = "测量完成";
public static final String ECG_TYPE_MEASURE_FAILURE = "测量失败";
```

String errorType,是测量过程中心电图有异常返回的异常值,他的值是 EcgResolve 中的常量

```
public static final String ECG_ERROR_TYPE1 = "正常";
public static final String ECG_ERROR_TYPE2 = "波形疑似心跳稍快";
public static final String ECG_ERROR_TYPE3 = "波形疑似心跳过快";
public static final String ECG_ERROR_TYPE4 = "波形疑似阵发性心跳过快";
public static final String ECG_ERROR_TYPE5 = "波形疑似心跳稍缓";
public static final String ECG_ERROR_TYPE6 = "波形疑似心跳过缓";
public static final String ECG_ERROR_TYPE7 = "波形疑似偶发心跳间期缩短";
public static final String ECG_ERROR_TYPE8 = "波形疑似心跳稍缓, 伴有偶发心跳间期缩短";
public static final String ECG_ERROR_TYPE9 = "波形疑似心跳稍快, 伴有偶发心跳间期缩短";
public static final String ECG_ERROR_TYPE10 = "波形疑似心跳稍缓, 伴有偶发心跳间期缩短";
public static final String ECG_ERROR_TYPE11 = "波形疑似心跳稍缓, 伴有偶发心跳间期不规则";
public static final String ECG_ERROR_TYPE12 = "波形有漂移";
public static final String ECG_ERROR_TYPE13 = "波形疑似心跳过快, 伴有漂移";
public static final String ECG_ERROR_TYPE14 = "波形疑似心跳过缓, 伴有漂移";
public static final String ECG_ERROR_TYPE15 = "波形疑似偶发心跳间期缩短, 伴有漂移";
public static final String ECG_ERROR_TYPE16 = "波形疑似心跳间期不规则, 伴有漂移";
```

List<Integer> listWave, 是心电图的波形, 波形范围请参考美的连设备协议中的范围, 需要使用自定义 View, 将该集合中的点连成线形成心电图

Int ecgRate; 是心率值,在稳定测量一定时间后,会出现心率值

## 4.IrtResolve 体温计解析

String type, 是测量进行到了哪一步, 他的值是 IrtResolve 中的常量

```
public static final String IRT_TYPE_NOT_CONNECT = "未连接";
public static final String IRT_TYPE_READY = "设备就绪";
public static final String IRT_TYPE_MEASURE = "测量中...";
public static final String IRT_TYPE_MEASURED = "测量完成";
public static final String IRT_TYPE_MEASURE_FAILURE = "测量失败";
```

float tmperature,测量完成之后返回的摄氏度

## 5.NibpResolve 血压计解析

String type, 是测量进行到了哪一步, 他的值是 NibpResolve 中的常量

```
public static final String NIBP_TYPE_NOT_CONNECT = "未连接";
public static final String NIBP_TYPE_READY = "准备中...";
public static final String NIBP_TYPE_MEASURE = "测量中...";
public static final String NIBP_TYPE_MEASURED = "测量完成";
public static final String NIBP_TYPE_MEASURE_FAILURE = "测量失败";
public static final String NIBP_TYPE_MEASURE_STOP = "测量停止";
```

int pressure, 状态处于测量中时传回来的气压值, 需要在 UI 中实时显示当前的气压值

int sbp, 测量完成后传回来的收缩压

int dbp, 测量完成后传回来的舒张压

int pulse, 测量完成后传回来的脉率值

String pulseIrregularity, 测量完成后传回来的心率不齐标志, 参数为以下常量

```
public static final String PULSE_IRREGULARITY_SUCCESS = "心率正常";  
public static final String PULSE_IRREGULARITY_FAILURE = "心率不齐";
```

String errorType, 测量错误返回的错误类型, 参数为以下常量

```
public static final String NIBP_TYPE_00 = " ";  
public static final String NIBP_TYPE_01 = "打气 11s 袖带气压不足 50mmHg";  
public static final String NIBP_TYPE_02 = "气袋压力超过 295mmHg";  
public static final String NIBP_TYPE_03 = "测量不到有效脉搏";  
public static final String NIBP_TYPE_04 = "干扰过多";  
public static final String NIBP_TYPE_05 = "测量结果数值有误";
```

## 6.Spo2PackageData 血氧解析

血氧的 Spo2PackageData 中的数据

```
//连接标志: bit5=1 未连接传感器,=0 为已连接  
private boolean connectMark;//连接标志  
//搜索标志: bit4=1, 搜索时间太长  
private boolean searchMark;//搜索标志  
private int rssi;//信号强度  
private int pulse;//脉率  
private int breathe;//呼吸  
private int box;//血氧值  
private float temperature;//温度  
private float pi;//灌注度
```

### 6.1 Spo2PackageWave 血氧容积脉搏波

血氧的 Spo2PackageData 中的数据

List<Integer> spo2Wave,此处与心电图类似, 根据每个点绘制自定义 View, 形成血氧容积脉搏波

List<String> waveStatus,波形的异常状态,以及脉搏音,如果当中值有脉搏音时,则让 Android 设备发出脉搏音的响声,其余错误则提示用户, 以下是他的值

```
public static final String WAVE_STATUS_NORMAL = "正常";  
public static final String WAVE_STATUS_FINDING = "正在搜索脉搏";
```

```
public static final String WAVE_STATUS_SENSOR_ERROR = "传感器错误";
public static final String WAVE_STATUS_PULES = "脉搏音";
//波形强度
private List<Integer> pulseLevel = new ArrayList<>();
```

List<String> pulseLevel, 波形的强度范围值是 0-15

## 6.2 Spo2PackageHRV 血氧容积脉搏波

List<Integer> hrvList, HRV 是指逐次心跳周期差异的变化情况

## 7.EcgResolve2 心电解析(读心宝)

String type, 是测量进行到了哪一步, 他的值是 EcgResolve 中的常量

```
public static final String ECG_TYPE_NOT_CONNECT = "未连接";
public static final String ECG_TYPE_READY = "设备就绪";
public static final String ECG_TYPE_MEASURE = "测量中...";
public static final String ECG_TYPE_MEASURED = "测量完成";
public static final String ECG_TYPE_MEASURE_FAILURE = "测量失败";
```

String errorType, 是测量过程中心电图有异常返回的异常值, 他的值是 EcgResolve 中的常量

```
public static final String ECG_ERROR_TYPE1 = "正常";
public static final String ECG_ERROR_TYPE2 = "波形疑似心跳稍快";
public static final String ECG_ERROR_TYPE3 = "波形疑似心跳过快";
public static final String ECG_ERROR_TYPE4 = "波形疑似阵发性心跳过快";
public static final String ECG_ERROR_TYPE5 = "波形疑似心跳稍缓";
public static final String ECG_ERROR_TYPE6 = "波形疑似心跳过缓";
public static final String ECG_ERROR_TYPE7 = "波形疑似偶发心跳间期缩短";
public static final String ECG_ERROR_TYPE8 = "波形疑似心跳稍缓, 伴有偶发心跳间期缩短";
public static final String ECG_ERROR_TYPE9 = "波形疑似心跳稍快, 伴有偶发心跳间期缩短";
public static final String ECG_ERROR_TYPE10 = "波形疑似心跳稍缓, 伴有偶发心跳间期缩短";
public static final String ECG_ERROR_TYPE11 = "波形疑似心跳稍缓, 伴有偶发心跳间期不规则";
public static final String ECG_ERROR_TYPE12 = "波形有漂移";
public static final String ECG_ERROR_TYPE13 = "波形疑似心跳过快, 伴有漂移";
public static final String ECG_ERROR_TYPE14 = "波形疑似心跳过缓, 伴有漂移";
public static final String ECG_ERROR_TYPE15 = "波形疑似偶发心跳间期缩短, 伴有漂移";
public static final String ECG_ERROR_TYPE16 = "波形疑似心跳间期不规则, 伴有漂移";
```

List<Integer> listStartWave, 是心电图的波形, 波形范围请参考美的连设备协议中的范围, 需要使用自定义 View, 将该集合中的点连成线形成心电图

Int ecgRate; 是心率值, 在稳定测量一定时间后, 会出现心率值

