METTLER TOLEDO安卓传秤接口说明

|  |  |
| --- | --- |
| 版本 | 变更说明 |
| 1.0 | 初版，支持PLUET |
| 1.1 | 增加预置键、条码、标签格式 |

本接口运行于Android平台，以jar包形式供第三方调用，从而实现向bPlus写入商品数据。目前不支持远程取数和流水回收。

1. 包名：com.mt.rt.aoapi
2. 需要的权限

<uses-permission android:name="android.permission.INTERNET" />

<uses-permission android:name="android.permission.READ\_EXTERNAL\_STORAGE" />

1. 调用接口

首先创建AOScale类的实例，参数为Handler实例（用于接收结果消息，详见示例代码）

AOScale mAOScale = new AOScale(handler);

然后调用AOScale的write方法，将JSON数据写入指定的电子秤列表：

void write(java.util.List<android.util.Pair<String, Integer>> scales,  
 com.mt.rt.aoapi.DataType type,  
 org.json.JSONObject data);

其中参数为：

* scales：电子秤列表，List中的每个元素为IP地址和端口号组成的Pair，**List长度不可大于8**
* type：枚举值表示传输的数据类型，目前支持DataType.PLUET（商品数据）、DataType.BARCODE（条码格式）、DataType.PRESETKEY（预置键）、DataType.LABEL（标签格式）。
* data：JSON数据，定义见后文；一次传输的商品不能太多，详见示例代码。

1. 获取结果

可以在上述Handler的handleMessage方法中，处理各种消息，以获取结果。详见后文示例代码。

1. 输入的JSON定义，红色是必填项
   1. 商品信息

|  |  |
| --- | --- |
| { |  |
| "data": [{ |   JSONArray，即商品可以是多个，但不能太多，详见示例代码 |
| "plu": "1", |   PLU号 |
| "descriptions": ["2级牛肉"], |   商品名称，只会解析JSONArray中的第一个元素 |
| "priceA": { |   单价，只支持第一单价 |
| "value": 88.68, |   单价值 |
| "uom": "PCS", |   计价方式，可以是GRM、KGM、PCS，分别表示g、kg、计数 |
| "unitDes": "包", |   计数商品单位，**暂不支持此字段** |
| "discount": false, |   允许打折，不填则默认false |
| "override": false, |   允许改价，不填则默认false |
| "type": "BasePrice" |   填写固定的BasePrice |
| }, |  |
| "department": "1", |   部门号 |
| "group": "1", |   分组号 |
| "categories": ["0"], |   分类号，最多填写5个，一般只用1个 |
| "tare": { |  |
| "value": 2, |   商品皮重值 |
| "uom": "KGM" |   商品皮重单位，和上述计价方式相同 |
| }, |  |
| "packed": { |  |
| "value": "2014-09-03", |   包装日期 |
| "unit": "date", |   包装日期单位 |
| "printable": true |   包装日期打印标记 |
| }, |  |
| "bestBefore": { |  |
| "value": "1", |   推荐日期 |
| "unit": "day", |   推荐日期单位 |
| "printable": true |   推荐日期打印标记 |
| }, |  |
| "sellBy": { |  |
| "value": "2", |   保质日期 |
| "unit": "day", |   保质日期单位 |
| "printable": true |   保质日期打印标记 |
| }, |  |
| "barcode": 12, |   条码号 |
| "labels": [1, 2, 3], |   标签号，一般只填一个值 |
| "et1": "牛肉", |   附加文本用于配料表、产地等附加信息。附加文本段落1 |
| "et2": "蛋白质", |   附加文本段落2 |
| "et3": "脂肪", |   附加文本段落3 |
| "et4": "水", |   附加文本段落4 |
| "et5": "铁", |   附加文本段落5 |
| "et6": "金坛雀舌", |   附加文本段落6 |
| "et7": "溧阳白茶", |   附加文本段落7 |
| "et8": "毛尖" |  附加文本段落8 |
| }] |  |
| } |  |

* 1. 条码

|  |  |
| --- | --- |
| { |  |
| "barcodes": [{ |   JSONArray |
| "id": "1", |   条码号 |
| "type": "EAN13", |   条码类型 |
| "description": "1", |   条码描述 |
| "definition": "24PPPPPBBBBBC" |   条码格式定义 |
| }, |  |
| { |  |
| "id": "21", |  |
| "type": "QRCode", |  |
| "description": "adv barcode", |  |
| "definition": "$R(Registration -Separator 13[$D(ItemDescription -l 10)$D(TransactionNetWeight -l 5)$D(TransactionUnitPrice -l 5)$D(TransactionTotalPrice -l 5)])" |  |
| } |  |
| ] |  |
| } |  |

* 1. 预置键

|  |  |
| --- | --- |
| { |  |
| "page": { |  |
| "h": 7, |   页面行数，不填默认为7 |
| "w": 10 |   页面列数，不填默认为10 |
| }, |  |
| "keys": [{ |   JSONArray |
| "type": "PLU", |   类型填写"PLU" |
| "location": [1,1,1], |   目标键位，三个值分别表示层数、行数、列数，层数只能是1或者2 |
| "dest": "60", |   设置的PLU号 |
| "title": "" |   title字段忽略，可不填 |
| }, |  |
| { |  |
| "type": "PLU", |  |
| "location": [1,5,2], |  |
| "dest": "None(0)", |   填写"None(0)"可将该键位清空 |
| "title": "" |  |
| }, |  |
| { |  |
| "type": "PLU", |  |
| "location": [2,3,6], |  |
| "dest": "58", |  |
| "title": "" |  |
| } |  |
| ] |  |
| } |  |

* 1. 标签格式

|  |  |
| --- | --- |
| { |  |
| "labels": { |  |
| "path": "/mnt/sdcard/mttest/LBL00101.xml" |   标签文件的路径，该文件是由SmartLabel导出的 |
| } |  |
| } |  |

1. 获取日志。本接口的日志的tag均为MTCT

adb logcat -s MTCT

1. 示例代码

AOScale mAOScale = new AOScale(handler);

List<Pair<String, Integer>> scales = …;  
JSONObject json = …;  
  
if (!mAOScale.isWriteBusy()) {  
 mAOScale.write(scales, DataType.PLUET, json);  
}

*// 在合适的地方调用cleanup*mAOScale.cleanup();

*// handler用于创建*AOScale*实例，实现*handleMessage方法处理结果消息private Handler handler = new Handler() {  
 @Override  
 public void handleMessage(Message msg) {  
 switch (msg.what) {  
 case AOScale.*EVT\_COMM\_TRANSMITION\_BEGIN*:  
 *// 开始向一台电子秤写入数据* break;  
 case AOScale.*EVT\_COMM\_TRANSMITION\_END*:  
 *// 结束向一台电子秤写入数据* break;  
 case AOScale.*EVT\_COMM\_WAITE\_ALL\_END*:  
 *// 向所有电子秤的写入均已结束* *// 调用如下函数获取所有的结果* List<Integer> retList = mAOScale.getWriteErrorCode();  
 break;  
 case AOScale.*EVT\_COMM\_WAITE\_DATA\_TOOLONG*:  
 *// 待传输的数据太多，应减少后重试* break;  
 case AOScale.*EVT\_COMM\_SCALE\_REPORT\_ERROR*:  
 *// 电子秤回应错误* break;  
 case AOScale.*EVT\_COMM\_SENDDATA\_SECCESS*:  
 *// 成功向电子秤写入数据* break;  
 case AOScale.*EVT\_COMM\_SENDDATA\_EXCEPTION*:  
 *// 向电子秤写入数据时出错（捕获到异常），可以检查日志查找错误原因* break;  
 case AOScale.*EVT\_COMM\_RECVDATA\_UNCOMPLETE*:  
 *// 读取到的电子秤响应数据不完整* break;  
 case AOScale.*EVT\_COMM\_RECVDATA\_UNKNOWN*:  
 *// 读取到的电子秤响应数据不可识别* break;  
 case AOScale.*EVT\_COMM\_RECVDATA\_EXCEPTION*:  
 *// 读取电子秤响应数据时出错（捕获到异常），可以检查日志查找错误原因* break;  
 case AOScale.*EVT\_COMM\_RECVDATA\_TIMEOUT*:  
 *// 读取电子秤响应数据时超时* break;  
 case AOScale.*EVT\_SOCKET\_OPEN\_FAILED*:  
 *// 初始化和电子秤的连接失败* break;  
 case AOScale.*EVT\_SOCKET\_CONNECT\_FAILED*:  
 *// 连接电子秤失败* break;  
 case AOScale.*EVT\_SOCKET\_CONNECT\_EXCEPTION*:  
 *// 连接电子秤时出错（捕获到异常），可以检查日志查找错误原因* break;  
 case AOScale.*EVT\_SOCKET\_DISCONNECT\_EXCEPTION*:  
 *// 断开和电子秤的连接时出错（捕获到异常），可以检查日志查找错误原因* break;  
 case AOScale.*EVT\_PARSE\_JSON\_EXCEPTION*:  
 *// 解析传入的JSON数据时出错，可以检查日志查找错误原因* break;  
 case AOScale.*EVT\_GENERAL\_EXCEPTION*:  
 *// 捕获到其它异常，可以检查日志查找错误原因* break;  
 case AOScale.*EVT\_SCALE\_COUNT\_IS\_FULL*:  
 *// 同时传输的电子秤太多，应减少参数中的电子秤数量后重试* break;  
 default:  
 break;  
 }  
  
 super.handleMessage(msg);  
 }  
};