RFID开发文档

V2.3

**文档控制**

更改记录

|  |  |  |  |
| --- | --- | --- | --- |
| **日期** | **作者** | **版本** | **更改记录** |
| 2022-08-25 | 唐腾 | v1.1 | Api网页文档转为中文SDK文档 |
| 2022-09-08 | 王荣超 | v1.2 | 新增6B标签接口 |
| 2022-10-30 | 王荣超 | v1.3 | 盘存标签新增回调方法 onInventory() |
| 2022-11-23 | 唐腾 | V2.0 | 删除一些api和新增一些常用api，同时更新盘点相关操作流程。 |
| 2022-12-16 | 唐腾 | V2.1 | 修改盘点、设置盘点参数、setTagMask、clearTagMask方法。弃用获取盘点参数方法 |
| 2023-1-4 | 唐腾 | V2.2 | 新增设置自定义频段方法 |
| 2023-2-1 | 唐腾 | V2.3 | 新设置掩码，设置链路，通过tid读取标签数据。短距支持盘点的时候携带tid。 |

**目录**

[1. 常用方法使用流程 4](#_Toc8697)

[1.1 初始化 4](#_Toc7128)

[1.2 盘点 4](#_Toc17928)

[1.3 读取标签 4](#_Toc11006)

[1.4 写标签 5](#_Toc4283)

[1.5 读取功率 5](#_Toc9241)

[1.6 写入功率 5](#_Toc21740)

[2. 接口详情 5](#_Toc6045)

[回调接口 5](#_Toc20028)

[2.1 6C回调接口的方法详情。 5](#_Toc29625)

[2.2 读取6B标签回调接口的方法详情。 6](#_Toc18772)

[3. RFID相关接口 8](#_Toc11771)

[3.1 init【初始化】 8](#_Toc4805)

[3.2 release【释放资源】 8](#_Toc26652)

[3.3 sendCommand【发送指令】 8](#_Toc26918)

[3.4 disConnect【断开rfid连接】 9](#_Toc32328)

[3.5 isConnected【判断是否连接上rfid模块】 9](#_Toc25554)

[3.6 cancelAccessEpcMatch【清除EPC匹配】 9](#_Toc15184)

[3.7 clearTagMask【清除标签掩码】 10](#_Toc30640)

[3.8 startInventory【盘点】 10](#_Toc10359)

[3.9 stopInventory【停止标签存货】 10](#_Toc6434)

[3.10 getAccessEpcMatch【查询并匹配epc状态】 10](#_Toc24593)

[3.11 getFrequencyRegion【获取频率区域】 11](#_Toc6816)

[3.12 getFirmwareVersion【获取固件版本】 11](#_Toc28235)

[3.13 getIdentifier【获取验证者】 11](#_Toc5364)

[3.14 getInventoryBuffer【获取标签数据缓存】 11](#_Toc30953)

[3.15 getInventoryBufferTagCount【获取标签在缓存中的数量】 12](#_Toc20073)

[3.16 getOutputPower【查询输出功率】 12](#_Toc28221)

[3.17 getReaderIdentifier【获得阅读器标识符】 12](#_Toc115)

[3.18 getReaderTemperature【查询内部温度】 12](#_Toc15534)

[3.19 iso180006BInventory【盘存 18000-6B 标签】 13](#_Toc6039)

[3.20 iso180006BLockTag【锁住 18000-6B 标签】 13](#_Toc24444)

[3.21 iso180006BQueryLockTag【查询 锁住的18000-6B 标签】 13](#_Toc10144)

[3.22 iso180006BReadTag【读18000-6B 标签】 14](#_Toc12970)

[3.23 iso180006BWriteTag【写18000-6B 标签】 14](#_Toc24159)

[3.24 killTag【杀死标签】 14](#_Toc20275)

[3.25 lockTag【锁住标签】 15](#_Toc20876)

[3.26 readTag【读取标签TID】 15](#_Toc31224)

[3.27 realTimeInventory【盘存标签（读取时间模式）】 16](#_Toc10363)

[3.28 rest【重置指定地址阅读器】 16](#_Toc17162)

[3.29 setAccessEpcMatch【设置访问 EPC 匹配】 16](#_Toc26826)

[3.30 setFrequencyRegion【设置频率区域】 17](#_Toc8241)

[3.31 setOutputPower【设置输出功率】 17](#_Toc9525)

[3.32 setTagMask【设置标签掩码】 18](#_Toc25839)

[3.33 setTrigger【设置触发器】 19](#_Toc2084)

[3.34 writeTag【写标签】 19](#_Toc19273)

[3.35 registerCallback【注册回调】 19](#_Toc16321)

[3.36 unregisterCallback【注销回调】 20](#_Toc7190)

[3.37 getModule【获取模块】 20](#_Toc16269)

[3.38 getModuleFirmware【获取模块固件信息】 20](#_Toc15027)

[3.39 writeEpc【广播方式随机改写一张标签的EPC号】 20](#_Toc11215)

[3.40 readOnceTag【单标签查询】 21](#_Toc14142)

[3.41 writeTagByTid【通过tid向各存储区写数据】 21](#_Toc9559)

[3.42 setScanInterval【设置盘存间隔时间】 22](#_Toc25548)

[3.43 getScanInterval【读取盘存间隔时间】 22](#_Toc8289)

[3.44 setQueryMode【设置查询模式】 22](#_Toc3127)

[3.45 getQueryMode【获取查询模式】 22](#_Toc1958)

[3.46 GetReaderType【获取设备是长距还短距设备】 23](#_Toc11638)

[3.47 startRead【启动盘点】 23](#_Toc13530)

[3.48 scanRfid【询查标签】 23](#_Toc12449)

[3.49 setInventoryParameter【设置盘点参数】 23](#_Toc21029)

[3.50 getInventoryParameter【获取盘点参数】注：该方法已经弃用 24](#_Toc3843)

[3.51 getDeviceId【获取序列号】 24](#_Toc5208)

[3.52 setCustomRegion【设置自定义频段】（国标无效） 24](#_Toc12278)

[3.53 getCustomRegion【获取自定义频段】（国标无效） 25](#_Toc31735)

[3.54 readDataByTid【通过tid读取数据】 25](#_Toc5718)

[3.55 addMask【添加掩码】（短距不支持） 25](#_Toc31556)

[3.56 clearMask【清除掩码】 26](#_Toc25291)

[3.57 setProfile【设置链路配置信息】 26](#_Toc4593)

[3.58 setRange【设置识别距离】 27](#_Toc6468)

[3.59 getRange【获取设置的识别距离】 27](#_Toc13096)

[常见错误码 27](#_Toc19282)

1. **常用方法使用流程**
   1. **初始化**

|  |
| --- |
| **private void initRfid() {**  **// 在异步回调中拿到RFID实例**  **USDKManager.getInstance().init(BaseApplication.getContext(),new USDKManager.InitListener() {**  **@Override**  **public void onStatus(USDKManager.STATUS status) {**  **if ( status == USDKManager.STATUS.SUCCESS) {**  **Log.d(TAG, "initRfid() success.");**  **mRfidManager = USDKManager.getInstance().getRfidManager();**  **}else {**  **Log.d(TAG, "initRfid fail.");**  **}**  **}**  **});**  **}** |

* 1. **盘点**

当rfidmanager初始化后就可以使用盘点方法。

通过rfidmanager.customizedSessionTargetInventory方法就可以开始盘点。盘点的结果可以通过，自己实现的IRfidCallback 的实现类中onInventoryTag获取相关数据。

**步骤：**

1. 开始盘点：

Rfidmanager.customizedSessionTargetInventory(btSession,btTarget,btRepeat);

1. 回调实现体中的onInventoryTag(回调)
   1. **读取标签**

通过RFIDManager的readTag方法来读标签，

步骤：

1. 读取标签：

String result=

rfidmanager.readTag(epc,btMemBank,btWordAdd,btWordCnt,btAryPassWord),直接返回标签读到的结果。

* 1. **写标签**

通过rfidmanager的writeTag方法。

步骤：

1. 写标签：rfidmanager.writeTag(epc,btAryPassWord,btMemBank,btWordAdd,btWordCnt,btAryData),该方法会直接返回结果，0是成功，其他是失败。

**1.5 读取功率**

通过rfidmanager的getOutputPower获取，

1. 获取功率：rfidmanager.getOutputPower()，该方法会直接返回当前设备的功率。

**1.6 写入功率**

通过rfidmanager的setOutputPower方法。

1. 设置输出功率：rfidmanager.setOutputPower(power),该方法会直接返回int值0是成功，其他是失败。
2. **接口详情**

**回调接口**

public interface IRfidCallback {

void onInventoryTag(String EPC, String TID, String strRSSI) ;

void onInventoryTagEnd();

}

**2.1 6C回调接口的方法详情。**

注：下面的函数的参数名和上面是一一对应的，只是给出了具体含义，有一些参数目前没有实际作用。

onInventoryTag

|  |  |  |  |
| --- | --- | --- | --- |
| **定义** | void onInventoryTag(String EPC, String TID, String strRSSI); | | |
| **说明** | 该方法会在开启盘点后，通过回调该方法返回找到标签的信息。 | | |
| **参数** | **名称** | **类型** | **备注** |
| EPC | String | 盘点到的标签的epc号 |
| TID | String | 标签的TID |
| strRSSI | String | 标签的信号强度 |

**onInventoryTagEnd**

|  |  |
| --- | --- |
| **定义** | void onInventoryTagEnd(); |
| **说明** | 此方法盘点结束的时候调用。 |

### 2.2 读取6B标签回调接口的方法详情。

ITag6BCallback {

public void onRead6BTag( String strData) ;

public void onWrite6BTag( byte nWriteLen) ;

public void onLock6BTag(byte nStatus) ;

public void onLockQuery6BTag( byte nStatus) ;

public void onExeCMDStatus(byte cmd, byte status);

public void onInventory6BTag( String strUID );

}

#### **onRead6BTag**

|  |  |  |  |
| --- | --- | --- | --- |
| 定义 | void onRead6BTag( String strData); | | |
| 说明 | 读取6B标签的结果回调 | | |
| 参数 | 名称 | 类型 | 备注 |
| strData | String | 读取到的6B标签 |

#### **onWrite6BTag**

|  |  |  |  |
| --- | --- | --- | --- |
| 定义 | void onWrite6BTag( byte nWriteLen); | | |
| 说明 |  | | |
| 参数 | 名称 | 类型 | 备注 |
| nWriteLen | byte | （目前版本已经不回调该方法） |

#### **onLock6BTag**

|  |  |  |  |
| --- | --- | --- | --- |
| 定义 | void onLock6BTag( byte nStatus); | | |
| 说明 | 锁定6B标签的结果回调 | | |
| 参数 | 名称 | 类型 | 备注 |
| nStatus | byte | 0-锁定成功 1-锁定失败 |

#### **onLockQuery6BTag**

|  |  |  |  |
| --- | --- | --- | --- |
| 定义 | void onLockQuery6BTag( byte nStatus); | | |
| 说明 | 查询6B标签的锁定状态 | | |
| 参数 | 名称 | 类型 | 备注 |
| nStatus | byte | 0-未被锁定；1-被锁定 |

#### **onInventory6BTag**

|  |  |  |  |
| --- | --- | --- | --- |
| 定义 | void onInventory6BTag( String strUID ); | | |
| 说明 | 盘存到 6B标签的标签值 | | |
| 参数 | 名称 | 类型 | 备注 |
| strUID | String | 盘存到的6B标签值 |

#### **onExeCMDStatus**

|  |  |  |  |
| --- | --- | --- | --- |
| 定义 | void onExeCMDStatus(byte cmd, byte result); | | |
| 说明 | 指令执行情况。读写设置参数指令都会调用这个方法 | | |
| 参数 | 名称 | 类型 | 备注 |
| cmd | byte | 操作的命令 |
| result | byte | 命令执行的结果 |

1. **RFID相关接口**

3.1 init【初始化】

|  |  |  |  |
| --- | --- | --- | --- |
| **定义** | public void init(); | | |
| **说明** | 初始化RFIDManager。 | | |
| **参数** | **名称** | **类型** | **备注** |
|  |  |  |
| **返回** | 无 | | |
| **参考代码** | 无。 | | |

3.2 release【释放资源】

|  |  |  |  |
| --- | --- | --- | --- |
| **定义** | public void release(); | | |
| **说明** | 释放资源 | | |
| **参数** | **名称** | **类型** | **备注** |
|  |  |  |
| **返回** | 无 | | |
| **参考代码** | 无 | | |

3.3 sendCommand【发送指令】

|  |  |  |  |
| --- | --- | --- | --- |
| **定义** | public int sendCommand(byte[] btCMDPackage); | | |
| **说明** | 发送指令 | | |
| **参数** | **名称** | **类型** | **备注** |
| btCMDPackage | byte[] | 指令 |
| **返回**  **(int)** | 成功：0,失败：-1； | | |
| **参考代码** | 无 | | |

3.4 disConnect【断开rfid连接】

|  |  |  |  |
| --- | --- | --- | --- |
| **定义** | public void disConnect(); | | |
| **说明** | 断开rfid连接模块。 | | |
| **参数** | **名称** | **类型** | **备注** |
|  |  |  |
| **返回**  **(void)** |  | | |
| **参考代码** | 无 | | |

3.5 isConnected【判断是否连接上rfid模块】

|  |  |  |  |
| --- | --- | --- | --- |
| **定义** | public boolean isConnected(); | | |
| **说明** | 判断是否连接上。 | | |
| **参数** | **名称** | **类型** | **备注** |
|  |  |  |
| **返回**  **(boolean)** | 成功：true ；失败：false ； | | |
| **参考代码** | 无 | | |

3.6 cancelAccessEpcMatch【清除EPC匹配】

|  |  |  |  |
| --- | --- | --- | --- |
| **定义** | public int cancelAccessEpcMatch(); | | |
| **说明** | 清除EPC匹配。 | | |
| **参数** | **名称** | **类型** | **备注** |
| **返回**  **(int)** | 成功：0；失败：-1； | | |
| **参考代码** | 无 | | |

3.7 clearTagMask【清除标签掩码】

|  |  |  |  |
| --- | --- | --- | --- |
| **定义** | public int clearTagMask(); | | |
| **说明** | 清除标签掩码。 | | |
| **参数** | **名称** | **类型** | **备注** |
| **返回**  **(int)** | 成功：0；失败：-1； | | |
| **参考代码** | 无 | | |

3.8 startInventory【盘点】

|  |  |  |  |
| --- | --- | --- | --- |
| **定义** | public int startInventory( byte time); | | |
| **说明** | 用户定义标签和存货。 | | |
| **参数** | **名称** | **类型** | **备注** |
| btSession | **byte** | **同一个id相同时间内读取的次数。** |
| **返回**  **(int)** | 成功：0；失败：-1； | | |
| **参考代码** | 无 | | |

3.9 stopInventory【停止标签存货】

|  |  |  |  |
| --- | --- | --- | --- |
| **定义** | public int stopInventory(); | | |
| **说明** | 停止读标签存货。 | | |
| **参数** | **名称** | **类型** | **备注** |
|  |  |  |
| **返回**  **(int)** | 成功：0；失败：-1； | | |
| **参考代码** | 无 | | |

3.10 getAccessEpcMatch【查询并匹配epc状态】

|  |  |  |  |
| --- | --- | --- | --- |
| **定义** | public int getAccessEpcMatch(); | | |
| **说明** | 查询匹配epc状态。 | | |
| **参数** | **名称** | **类型** | **备注** |
| **返回**  **(int)** | 成功：0；失败：-1； | | |
| **参考代码** | 无 | | |

3.11 getFrequencyRegion【获取频率区域】

|  |  |  |  |
| --- | --- | --- | --- |
| **定义** | public RfidDate getFrequencyRegion(); | | |
| **说明** | 获取频率区域。 | | |
| **参数** | **名称** | **类型** | **备注** |
| **返回**  **(**RfidDate **)** |  | | |
| **参考代码** | 无 | | |

3.12 getFirmwareVersion【获取固件版本】

|  |  |  |  |
| --- | --- | --- | --- |
| **定义** | public String getFirmwareVersion(); | | |
| **说明** | 获取读卡器固件版本。 | | |
| **参数** | **名称** | **类型** | **备注** |
| **返回**  **(String)** | 返回固件版本信息 | | |
| **参考代码** | 无 | | |

3.13 getIdentifier【获取验证者】

|  |  |  |  |
| --- | --- | --- | --- |
| **定义** | public byte[ ] getIdentifier(); | | |
| **说明** | 获取验证者。 | | |
| **参数** | **名称** | **类型** | **备注** |
|  |  |  |
| **返回**  **(byte[ ])** | 返回验证者地址数组 | | |
| **参考代码** | 无 | | |

3.14 getInventoryBuffer【获取标签数据缓存】

|  |  |  |  |
| --- | --- | --- | --- |
| **定义** | public int getInventoryBuffer(); | | |
| **说明** | 获取标签数据和保持缓存。 | | |
| **参数** | **名称** | **类型** | **备注** |
| **返回**  **(int)** | 成功：0；失败：-1； | | |
| **参考代码** | 无 | | |

3.15 getInventoryBufferTagCount【获取标签在缓存中的数量】

|  |  |  |  |
| --- | --- | --- | --- |
| **定义** | public int getInventoryBufferTagCount(); | | |
| **说明** | 获取标签在缓存中的数量。 | | |
| **参数** | **名称** | **类型** | **备注** |
| **返回**  **(int)** | 成功：0；失败：-1； | | |
| **参考代码** | 无 | | |

3.16 getOutputPower【查询输出功率】

|  |  |  |  |
| --- | --- | --- | --- |
| **定义** | public int getOutputPower(); | | |
| **说明** | 查询输出功率。 | | |
| **参数** | **名称** | **类型** | **备注** |
| **返回**  **(int)** | 返回输出功率 | | |
| **参考代码** | 无 | | |

3.17 getReaderIdentifier【获得阅读器标识符】

|  |  |  |  |
| --- | --- | --- | --- |
| **定义** | public int getReaderIdentifier(); | | |
| **说明** | 获得reader标识符。 | | |
| **参数** | **名称** | **类型** | **备注** |
| **返回**  **(byte)** | 成功：0；失败：-1； | | |
| **参考代码** | 无 | | |

3.18 getReaderTemperature【查询内部温度】

|  |  |  |  |
| --- | --- | --- | --- |
| **定义** | public RfidDate getReaderTemperature(); | | |
| **说明** | 查询内部温度。 | | |
| **参数** | **名称** | **类型** | **备注** |
| **返回**  **(RfidDate)** |  | | |
| **参考代码** | 无 | | |

3.19 iso180006BInventory【盘存 18000-6B 标签】

|  |  |  |  |
| --- | --- | --- | --- |
| **定义** | public int iso180006BInventory(); | | |
| **说明** | 盘存 18000-6B 标签。 | | |
| **参数** | **名称** | **类型** | **备注** |
| **返回**  **(int)** | 成功：0；失败：-1； | | |
| **参考代码** | 无 | | |

3.20 iso180006BLockTag【锁住 18000-6B 标签】

|  |  |  |  |
| --- | --- | --- | --- |
| **定义** | public int iso180006BLockTag(byte[] btAryUID,byte btWordAdd); | | |
| **说明** | 锁住 18000-6B 标签。 | | |
| **参数** | **名称** | **类型** | **备注** |
| btAryUID | **byte[]** | **操作标签的uid，8字节** |
| btWordAdd | **byte** | **锁住的地址** |
| **返回**  **(int)** | 成功：0；失败：-1； | | |
| **参考代码** | 无 | | |

3.21 iso180006BQueryLockTag【查询 锁住的18000-6B 标签】

|  |  |  |  |
| --- | --- | --- | --- |
| **定义** | public int iso180006BQueryLockTag(  byte[] btAryUID,  byte btWordAdd); | | |
| **说明** | 查询 锁住的18000-6B 标签。 | | |
| **参数** | **名称** | **类型** | **备注** |
| btAryUID | **byte[]** | **操作标签的uid，8字节** |
| btWordAdd | **byte** | **锁住的地址** |
| **返回**  **(int)** | 成功：0；失败：-1； | | |
| **参考代码** | 无 | | |

3.22 iso180006BReadTag【读18000-6B 标签】

|  |  |  |  |
| --- | --- | --- | --- |
| **定义** | public int iso180006BReadTag(  byte[] btAryUID,  byte btWordAdd,  byte btWordCnt); | | |
| **说明** | 读18000-6B 标签。 | | |
| **参数** | **名称** | **类型** | **备注** |
| btAryUID | **byte[]** | **操作标签的uid，8字节** |
| btWordAdd | **byte** | **数据首地址** |
| btWordCnt | **byte** | **数据长度** |
| **返回**  **(int)** | 成功：0；失败：-1； | | |
| **参考代码** | 无 | | |

3.23 iso180006BWriteTag【写18000-6B 标签】

|  |  |  |  |
| --- | --- | --- | --- |
| **定义** | public int iso180006BWriteTag(  byte[] btAryUID,  byte btWordAdd,  byte btWordCnt,  byte[] btAryBuffer); | | |
| **说明** | 读18000-6B 标签。 | | |
| **参数** | **名称** | **类型** | **备注** |
| btAryUID | **byte[]** | **操作标签的uid，8字节** |
| btWordAdd | **byte** | **数据首地址** |
| btWordCnt | **byte** | **数据长度** |
| btAryBuffer | **byte[]** | **待写入的数据** |
| **返回**  **(int)** | 成功：0；失败：-1； | | |
| **参考代码** | 无 | | |

3.24 killTag【杀死标签】

|  |  |  |  |
| --- | --- | --- | --- |
| **定义** | public int killTag(String epc,byte[] btAryPassWord); | | |
| **说明** | 杀死标签。 | | |
| **参数** | **名称** | **类型** | **备注** |
| epc | **String** | **标签的epc号** |
| btAryPassWord | **byte[]** | **清除标签密码，4字节。** |
| **返回**  **(int)** | 成功：0；失败：-1； | | |
| **参考代码** | 无 | | |

3.25 lockTag【锁住标签】

|  |  |  |  |
| --- | --- | --- | --- |
| **定义** | public int lockTag(String epc,  byte[] btAryPassWord,  byte btMemBank,  byte btLockType); | | |
| **说明** | 用于锁住标签。 | | |
| **参数** | **名称** | **类型** | **备注** |
| **Epc** | **String** | **标签的EPC** |
| btAryPassWord | **byte[]** | **进入密码，4字节。** |
| btMemBank | **byte** | **标签存储区（0x01：用户内存，0x03：TID内存，0x03：EPC内存，0x04：访问密码，0x05：Kill密码）** |
| btLockType | **byte** | **锁定操作类型（0x00：打开，0x01：锁定，0x03：永久打开，0x03：永久锁定）** |
| **返回**  **(int)** | 成功：0；失败：-1； | | |
| **参考代码** | 无 | | |

3.26 readTag【读取标签TID】

|  |  |  |  |
| --- | --- | --- | --- |
| **定义** | public String readTag(String EPC,  byte btMemBank,  byte btWordAdd,  byte btWordCnt,  byte[] btAryPassWord); | | |
| **说明** | 读取标签TID。 | | |
| **参数** | **名称** | **类型** | **备注** |
| EPC | **Stirng** | **标签的epc** |
| btAryPassWord | **byte[]** btWordCnt | **进入密码，4字节。** |
| btMemBank | byte | **标签存储区（0x00:RESERVED, 0x01:EPC, 0x03:TID, 0x03:USER）** |
| btWordAdd | **Byte[** | **开始读的地址** |
| btWordCnt | **byte** | **读取数据长度，以WORD(16bits)为单位的数据长度。** |
| **返回**  **(String)** | String :目标标签的tid | | |
| **参考代码** | 无 | | |

3.27 realTimeInventory【盘存标签（读取时间模式）】

|  |  |  |  |
| --- | --- | --- | --- |
| **定义** | public int realTimeInventory(byte btRepeat); | | |
| **说明** | 盘存标签（读取时间模式）。  注意：硬件为双CPU架构，主CPU负责标签盘点，副CPU负责数据管理。盘存和数据传输是并行和同步的。所以通过串口传输数据不会影响阅读器的效率。 | | |
| **参数** | **名称** | **类型** | **备注** |
| btRepeat | **byte** | **盘存轮次的重复时间。**  **当重复 = 355 时，盘存持续时间最小化。**  **例如，如果射频场只有一个或两个标签，盘点时间可能只有 30-50 毫秒，这个功能为设备上的快速天线切换应用提供了可能。** |
| **返回**  **(int)** | 成功：0；失败：-1； | | |
| **参考代码** | 无 | | |

3.28 rest【重置指定地址阅读器】

|  |  |  |  |
| --- | --- | --- | --- |
| **定义** | public int reset(); | | |
| **说明** | 重置指定地址阅读器。 | | |
| **参数** | **名称** | **类型** | **备注** |
| **返回**  **(int)** | 成功：0；失败：-1； | | |
| **参考代码** | 无 | | |

3.29 setAccessEpcMatch【设置访问 EPC 匹配】

|  |  |  |  |
| --- | --- | --- | --- |
| **定义** | public int setAccessEpcMatch(  byte btEpcLen,  byte[] btAryEpc); | | |
| **说明** | 设置访问 EPC 匹配（EPC 匹配有效，直到下次刷新）。 | | |
| **参数** | **名称** | **类型** | **备注** |
| btEpcLen | byte | **EPC长度** |
| btAryEpc | byte[] | **EPC，长度和btEpcLen相同** |
| **返回**  **(int)** | 成功：0；失败：-1； | | |
| **参考代码** | 无 | | |

3.30 setFrequencyRegion【设置频率区域】

|  |  |  |  |
| --- | --- | --- | --- |
| **定义** | public int setFrequencyRegion(  byte btRegion,  byte btStartRegion,  byte btEndRegion); | | |
| **说明** | 设置频率区域（系统默认频率）。 | | |
| **参数** | **名称** | **类型** | **备注** |
| btRegion | byte | **频谱规则（0x01：FCC，0x03：ETSI，0x03：CHN）。** |
| btStartRegion | byte | **频谱的起始频率。** |
| btEndRegion | byte | **频谱的终止频率**  **设置射频输出频谱的范围。**  **规则是：**  **1、起始频率和终止频率应在规定的规定范围内。**  **3、起始频率应等于或低于终止频率。**  **3、结束频率等于起始频率，即使用单频点。** |
| **返回**  **(int)** | 成功：0；失败：-1； | | |
| **参考代码** | 无 | | |

3.31 setOutputPower【设置输出功率】

|  |  |  |  |
| --- | --- | --- | --- |
| **定义** | public int setOutputPower(byte btOutputPower); | | |
| **说明** | 设置功率 | | |
| **参数** | **名称** | **类型** | **备注** |
| btOutputPower | byte | **射频输出功率：**  **新版固件功率范围：10-30**  **老版固件功率范围：0-33**  **短距版功率范围：0-26** |
| **返回**  **(int)** | 成功：0；失败：-1； | | |
| **参考代码** | 无 | | |

3.32 setTagMask【设置标签掩码】

|  |  |  |  |
| --- | --- | --- | --- |
| **定义** | public int setTagMask(  Int Membank,  Int StartAdd,  Int MaskLen,  String maskValue); | | |
| **说明** | 设置掩码过滤标签。 | | |
| **参数** | **名称** | **类型** | **备注** |
| Membank | int | **选择掩码区域、EPC、TID 或 USER。** |
| StartAdd | int | **掩码起始地址（按位）** |
| MaskLen | int | **掩码长度（按位）** |
| maskValue | String | **掩码值** |
| **返回**  **(int)** | 成功：0；失败：-1； | | |
| **参考代码** | 无 | | |

3.33 setTrigger【设置触发器】

|  |  |  |  |
| --- | --- | --- | --- |
| **定义** | public int setTrigger(boolean enable); | | |
| **说明** | 设置reader触发模式。 | | |
| **参数** | **名称** | **类型** | **备注** |
| enable | boolean | **启用操作reader** |
| **返回**  **(int)** | 成功：0；失败：-1； | | |
| **参考代码** | 无 | | |

3.34 writeTag【写标签】

|  |  |  |  |
| --- | --- | --- | --- |
| **定义** | public int writeTag(String epc,  byte[] btAryPassWord,  byte btMemBank,  byte btWordAdd,  byte btWordCnt,  byte[] btAryData); | | |
| **说明** | 写标签。 | | |
| **参数** | **名称** | **类型** | **备注** |
| **epc** | String | **标签的epc** |
| btAryPassWord | byte[] | **进入密码4字节** |
| btMemBank | byte | **标签存储区**  **（0x00:RESERVED,**  **0x01:EPC, 0x02:TID, 0x03:USER）。** |
| btWordAdd | byte | **写起始地址，WORD（16位）。**  **写 EPC 区时，注意 EPC 从地址 02 开始，前两个字为 PC+CRC。** |
| btWordCnt | byte | **16位的字** |
| btAryData | byte[] | **写入数据，btWordCnt\*2个字节** |
| **返回**  **(int)** | 成功：0；失败：-1； | | |
| **参考代码** | 无 | | |

3.35 registerCallback【注册回调】

|  |  |  |  |
| --- | --- | --- | --- |
| **定义** | public void registerCallback(com.ubx.usdk.rfid.aidl.IRfidCallback cb); | | |
| **说明** | 注册回调（标签数据或通知将在注册回调中回调）。 | | |
| **参数** | **名称** | **类型** | **备注** |
| cb | com.ubx.usdk.rfid.aidl.IRfidCallback | **回调** |
| **返回**  **(void)** | 无 | | |
| **参考代码** | 无 | | |

3.36 unregisterCallback【注销回调】

|  |  |  |  |
| --- | --- | --- | --- |
| **定义** | public void unregisterCallback(com.ubx.usdk.rfid.aidl.IRfidCallback cb); | | |
| **说明** | 注销回调。 | | |
| **参数** | **名称** | **类型** | **备注** |
| cb | com.ubx.usdk.rfid.aidl.IRfidCallback | **回调** |
| **返回**  **(void)** | 无 | | |
| **参考代码** | 无 | | |

3.37 getModule【获取模块】

|  |  |  |  |
| --- | --- | --- | --- |
| **定义** | public java.lang.String getModule(); | | |
| **说明** | 获取模块 | | |
| **参数** | **名称** | **类型** | **备注** |
| **返回**  **(String)** | 模块信息 | | |
| **参考代码** | 无 | | |

3.38 getModuleFirmware【获取模块固件信息】

|  |  |  |  |
| --- | --- | --- | --- |
| **定义** | public String getModuleFirmware(); | | |
| **说明** | 获取模块固件信息 | | |
| **参数** | **名称** | **类型** | **备注** |
| **返回**  **(String)** | 模块信息 | | |
| **参考代码** | 无 | | |

**3.39 writeEpc【广播方式随机改写一张标签的EPC号】**

|  |  |  |  |
| --- | --- | --- | --- |
| **定义1** | public int WriteEPC(byte epclen,byte epc[],byte Password[]); | | |
| **说明** | 广播方式随机改写一张标签的EPC号 | | |
| **参数** | **名称** | **类型** | **备注** |
| epclen | byte | 要写入的标签EPC号字长度 |
| epc | byte[] | 输入，标签的EPC号，大小为epclen\*2; |
| Password | byte[] | 输入，标签的访问密码，4个字节 |
| **返回**  **(int)** | 成功返回0。 | | |
| **参考代码** | 无 | | |

**3.40 readOnceTag****【单标签查询】**

|  |  |  |  |
| --- | --- | --- | --- |
| 定义 | public String readTagOnce( byte AdrTID, byte LenTID) | | |
| **说明** | 该命令用于单次读取标签的EPC或者TID。 | | |
| **参数** | **名称** | **类型** | **备注** |
| LenTID | byte | 读取TID长度，如果长度为0，则读取EPC号 |
|  | btReadId | byte | reader地址 |
| **返回**  **(String)** | 成功返回标签EPC或TID，失败返回null | | |
| **参考代码** | 无。 | | |

**3.41 writeTagByTid【通过tid向各存储区写数据】**

|  |  |  |  |
| --- | --- | --- | --- |
| **定义3** | public int writeTagByTid(String TIDStr,byte Mem,  byte WordPtr,byte Password[],String wdata); | | |
| **说明** | 向各存储区写数据. | | |
| **参数** | **名称** | **类型** | **备注** |
| TIDStr | String | 标签的tid号，16进制字符串，长度必须是4的整数倍 |
| Mem | byte | 待写入的存储区，   1. 密码区，前3个字是销毁密码，后3个字是访问密码 2. EPC区 3. TID区 4. 用户区 |
| WordPtr | int | 写入的起始字地址 |
| Password | String | 标签的访问密码，16进制字符串格式。 |
|  | WData | String | 待写入的数据，16进制字符串，长度必须是4的整数倍 |
| **返回**  **(int)** | 成功返回0。 | | |
| **参考代码** | 无 | | |

3.42 setScanInterval【设置盘存间隔时间】

|  |  |  |  |
| --- | --- | --- | --- |
| **定义** | public int setScanInterval( int interval) | | |
| **说明** | 设置启用盘点后使用的询查参数 | | |
| **参数** | **名称** | **类型** | **备注** |
| interval | int | 间隔时间，单位（毫秒） |
| **返回**  **(int)** | 成功返回0，失败返回-1 | | |
| **参考代码** | 无 | | |

3.43 getScanInterval【读取盘存间隔时间】

|  |  |  |  |
| --- | --- | --- | --- |
| **定义** | public int getScanInterval() | | |
| **说明** | 读取启用盘点后使用的询查参数 | | |
| **参数** | **名称** | **类型** | **备注** |
|  |  |  |
| **返回**  **(**int**)** | ReaderParameter：询查参数 | | |
| **参考代码** | 无 | | |

3.44 setQueryMode【设置查询模式】

|  |  |  |  |
| --- | --- | --- | --- |
| **定义** | public void setQueryMode(int mode) | | |
| **说明** | 可以通过三种方式来查询标签 | | |
| **参数** | **名称** | **类型** | **备注** |
| Mode | Int、 | 0：epc  1：epc+tid  2：epc+fasttid |
| **返回**  **(void)** |  | | |
| **参考代码** | 无 | | |

3.45 getQueryMode【获取查询模式】

|  |  |  |  |
| --- | --- | --- | --- |
| **定义** | public int getQueryMode() | | |
| **说明** | 获取设备的查询模式 | | |
| **参数** | **名称** | **类型** | **备注** |
| **返回**  **(int)** | 对应设置模式的值  0：epc  1：epc+tid  2：epc+fasttid | | |
| **参考代码** | 无 | | |

3.46 GetReaderType【获取设备是长距还短距设备】

|  |  |  |  |
| --- | --- | --- | --- |
| **定义** | public int GetReaderType() | | |
| **说明** | 获取设备的查询模式 | | |
| **参数** | **名称** | **类型** | **备注** |
| **返回**  **(int)** | 对应设置模式的值  80：短距  其他：长距 | | |
| **参考代码** | 无 | | |

3.47 startRead【启动盘点】

|  |  |  |  |
| --- | --- | --- | --- |
| **定义** | public int startRead() | | |
| **说明** | 启动盘点 | | |
| **参数** | **名称** | **类型** | **备注** |
| **返回**  **(int)** | 0：成功  -1：失败 | | |
| **参考代码** | 无 | | |

3.48 scanRfid【询查标签】

|  |  |  |  |
| --- | --- | --- | --- |
| **定义** | public void scanRfid() | | |
| **说明** | 开启盘点后一段时间自动停止 | | |
| **参数** | **名称** | **类型** | **备注** |
| **返回**  **(void)** |  | | |
| **参考代码** | 无 | | |

3.49 setInventoryParameter【设置盘点参数】

|  |  |  |  |
| --- | --- | --- | --- |
| **定义** | public void setInventoryParameter(Rfidparameter parameter) | | |
| **说明** | 设置盘点参数 | | |
| **参数** | **名称** | **类型** | **备注** |
| **返回**  **(void)** |  | | |
| **参考代码** | 无 | | |

3.50 getInventoryParameter【获取盘点参数】注：该方法已经弃用

|  |  |  |  |
| --- | --- | --- | --- |
| **定义** | public ReaderParameter getInventoryParameter() | | |
| **说明** | 获取盘点参数 | | |
| **参数** | **名称** | **类型** | **备注** |
| **返回**  **(**ReaderParameter **)** |  | | |
| **参考代码** | 无 | | |

3.51 getDeviceId【获取序列号】

|  |  |  |  |
| --- | --- | --- | --- |
| **定义** | public String getDeviceId() | | |
| **说明** | 获取序列号 | | |
| **参数** | **名称** | **类型** | **备注** |
| **返回**  **(String)** | 序列号rfid的序列号。 | | |
| **参考代码** | 无 | | |

3.52 setCustomRegion【设置自定义频段】（国标无效）

|  |  |  |  |
| --- | --- | --- | --- |
| **定义** | public int setCustomRegion(byte flags, int band, int FreSpace, int FreNum, int StartFre); | | |
| **说明** | 设置自定义频段 | | |
| **参数** | **名称** | **类型** | **备注** |
|  | flags | byte | **下电是否保存设置信息标识**  **0：保存**  **1：不保存** |
| int | band | **保留字段**  **固定为：0xff** |
| int | FreSpace | **频段间隔** |
| int | FreNum | **频点数量** |
| int | StartFre | **起始频段** |
| **返回**  **(int)** | 序列号rfid的序列号。  0:成功  其他:失败 | | |
| **参考代码** | 无 | | |

3.53 getCustomRegion【获取自定义频段】（国标无效）

|  |  |  |  |
| --- | --- | --- | --- |
| **定义** | public CustomRegionBean getCustomRegion(); | | |
| **说明** | 获取自定义频段信息，如果设置自定义频段后需用该方法来获取相关设置信息，调用getFrequencyRegion无法获取自定频段的相关信息。 | | |
| **参数** | **名称** | **类型** | **备注** |
| **返回**  **(**CustomRegionBean **)** | 序列号rfid的序列号。  CustomRegionBean 实例:成功  Null:失败  public class CustomRegionBean {  public int [] band=new int[1];//保留信息  public int [] FreSpace=new int[1];//频段间隔  public int [] FreNum=new int[1];//频点数量  public int [] StartFre=new int[1];//起始频点  public CustomRegionBean(){}  } | | |
| **参考代码** | 无 | | |

3.54 readDataByTid【通过tid读取数据】

|  |  |  |  |
| --- | --- | --- | --- |
| **定义** | public String readDataByTid(String TIDStr, byte Mem, byte WordPtr, byte Num, String Password) | | |
| **说明** | 通过标签的tid来读取数据 | | |
| **参数** | **名称** | **类型** | **备注** |
| TIDStr | String | **标签的tid** |
| Mem | byte | **读取的区域** |
| WordPtr | byte | **待读取数据的起始地址** |
| Num | byte | **需要读取数据的长度** |
| Password | String | **密码** |
| **返回**  **String** | Null：读取失败 | | |
| **参考代码** | 无 | | |

3.55 addMask【添加掩码】（短距不支持）

|  |  |  |  |
| --- | --- | --- | --- |
| **定义** | public void addMask(int mem,int startAddress,int len,String data) | | |
| **说明** | 添加盘点时候的过滤参数 | | |
| **参数** | **名称** | **类型** | **备注** |
| mem | int | **过滤区域** |
| startAddress | int | **起始地址** |
| len | int | **过滤长度** |
| data | String | **过滤数据** |
| **返回**  **Void** |  | | |
| **参考代码** | 无 | | |

3.56 clearMask【清除掩码】

|  |  |  |  |
| --- | --- | --- | --- |
| **定义** | public void clearMask(); | | |
| **说明** | 清除掩码数据 | | |
| **参数** | **名称** | **类型** | **备注** |
| **Void** |  | | |
| **参考代码** | 无 | | |

3.57 setProfile【设置链路配置信息】

|  |  |  |  |
| --- | --- | --- | --- |
| **定义** | public int setProfile(byte param); | | |
| **说明** | 设置链路信息 | | |
| **参数** | **名称** | **类型** | **备注** |
| param | byte | **一般这个参数的可选项需要根据设备信息来提供。**  **如果读取器的类型是**  **（0x21,0x23,0x28,0x36,0x37）**  **parm所代表的是：**  **0:40K, FM0,25us**  **1:250K,M4, 25us**  **2:300K,M4, 25us**  **3:400K,FM0,6.25us**  **如果读取器的类型不是上面那些类型**  **则param可供选项为一下**  **11:640K,FM0,7.5u**  **1:640K, M2,7.5us**  **15:640K, M4,7.5us**  **12:320K, M2, 15us**  **3:320K, M2, 20us**  **5:320K, M4, 20us**  **7:250K, M4, 20us**  **13:160K, M8, 20us** |
| **Void** |  | | |
| **参考代码** | 无 | | |

3.58 setRange【设置识别距离】

|  |  |  |  |
| --- | --- | --- | --- |
| **定义** | public int setRange(int range); | | |
| **描述** | 设置标签的识别距离 | | |
| **参数** | **name** | **Type** | **note** |
| Int | range | **0~100** |
| **Int** | 0:success; | | |
| Reference **Code** |  | | |

3.59 getRange【获取设置的识别距离】

|  |  |  |  |
| --- | --- | --- | --- |
| **定义** | public int getRange(); | | |
| **描述** | 获取设置的识别距离 | | |
| **参数** | **name** | **Type** | **note** |
|  |  |  |
| **返回值（int）** | -1:fail;  其他:设置的距离值 | | |
| 参考代码 |  | | |

**常见错误码**

|  |  |
| --- | --- |
| **错误代码** | **描述** |
| 0x00 | **执行成功。** |
| 0x01 | **未询查到电子标签。** |
| 0x05 | **访问密码错误。** |
| 0x09 | **销毁密码错误。** |
| 0x0A | **销毁密码不能为全0。** |
| 0x0B | **电子标签不支持该命令。** |
| 0x0C | **对该命令，访问密码不能为0。** |
| 0x0D | **电子标签已经被设置了读保护，不能再次设置。** |
| 0x0E | **电子标签没有被设置读保护，不需要解锁。** |
| 0x10 | **有字节空间被锁定，写入失败。** |
| 0x11 | **不能锁定。** |
| 0x12 | **已经锁定，不能再次锁定。** |
| 0x13 | **参数保存失败,但设置的值在读写模块断电前有效。** |
| 0x14 | **无法调整。** |
| 0xF8 | **天线检测错误** |
| 0xF9 | **命令执行出错。** |
| 0xFA | **有电子标签，但通信不畅，无法操作。** |
| 0xFB | **无电子标签可操作。** |
| 0xFC | **电子标签返回错误代码。** |
| 0xFD | **命令长度错误。** |
| 0xFE | **不合法的命令。** |
| 0xFF | **参数错误。** |
| 0x30 | **通讯错误。** |