CFBLELibrary接口函数说明

目录

[CFBLELibrary接口函数说明 1](#_Toc141348333)

[概述 2](#_Toc141348334)

[通用 2](#_Toc141348335)

[1. GetInstance 2](#_Toc141348336)

[2. StartScan 2](#_Toc141348337)

[3. ScanCHReader 3](#_Toc141348338)

[4. StopScan 3](#_Toc141348339)

[5. Connect 3](#_Toc141348340)

[6. DisConnect 5](#_Toc141348341)

[7. IsConnected 5](#_Toc141348342)

[设置 5](#_Toc141348343)

[1. Init 5](#_Toc141348344)

[2. GetInfo 6](#_Toc141348345)

[3. Reboot 6](#_Toc141348346)

[4. SetPower 7](#_Toc141348347)

[5. GetRFID 7](#_Toc141348348)

[6. SetRFID 8](#_Toc141348349)

[7. GetBattery 8](#_Toc141348350)

[8. GetBLEName 9](#_Toc141348351)

[9. SetBLEName 9](#_Toc141348352)

[10. GetAllParam 9](#_Toc141348353)

[11. SetAllParam 13](#_Toc141348354)

[12. GetBLEAddr 13](#_Toc141348355)

[盘点 14](#_Toc141348356)

[1. StartInventory 14](#_Toc141348357)

[2. StopInventory 15](#_Toc141348358)

[读写标签 16](#_Toc141348359)

[1. SelectTag 16](#_Toc141348360)

[2. ReadTag 16](#_Toc141348361)

[3. WriteTag 17](#_Toc141348362)

[4. LockTag 18](#_Toc141348363)

[5. KillTag 19](#_Toc141348364)

[附录A. 标签返回的操作状态 19](#_Toc141348365)

[附录B. STATUS的定义。 19](#_Toc141348366)

## 概述

CFBLELibrary是用于Adnroid系统下进行低功耗蓝牙应用开发的接口库，提供BLE设备的扫描、连接、检测、读写、订阅等API函数。

支持Android 5.1及以上系统，蓝牙控制器4.0及以上系统。

## 通用

### GetInstance

用于创建全局唯一实例

**public static** CFBLEManager GetInstance()

### StartScan

开始扫描附近的蓝牙设备

**public void** StartScan(ScanRuler ruler,@NonNull ScanObserver observer)

|  |  |
| --- | --- |
| 入参 |  |
| ScanRuler ruler | 扫描过滤规则 |
| @NonNull ScanObserver observer | 扫描回调 |
| 返回 | 无 |
| 抛出 | BLELibException |

* **参数说明：**

**ScanObserver :**

@Override

**public void** OnScanDevice(BluetoothDevice device, **int** rssi, **byte**[] broadcastRecord)

扫描到新设备时

|  |  |
| --- | --- |
| 入参 |  |
| BluetoothDevice device | 扫描到的设备 |
| **int** rssi |  |
| **byte**[] broadcastRecord |  |
| 返回 | 无 |

### ScanCHReader

开始扫描附近CH914X蓝牙设备

**public void** ScanCFReader(@NonNull ScanObserver observer)

|  |  |
| --- | --- |
| 入参 |  |
| @NonNull ScanObserver observer | 扫描回调 |
| 返回 | 无 |
| 抛出 | BLELibException |

* **参数说明：参见StartScan**

### StopScan

停止扫描

**public void** StopScan()

### Connect

连接蓝牙设备

**public void** Connect(ConnRuler ruler, @NonNull IConnectCallback callback)

|  |  |
| --- | --- |
| 入参 |  |
| ConnRuler ruler | 连接规则 |
| @NonNull IConnectCallback callback | 连接回调 |
| 返回 | 无 |
| 抛出 | BLELibException |

* **参数说明：**

**ConnRuler:**

ConnRuler connRuler=**new** ConnRuler.Builder(mac).connectTimeout(10000).build();

**IConnectCallback:**

@Override

**public void** OnError(String mac, Throwable t)

发生错误时

|  |  |
| --- | --- |
| 入参 |  |
| String mac | MAC码 |
| Throwable | 异常 |
| 返回 | 无 |

@Override

**public void** OnConnecting(String mac)

连接时

|  |  |
| --- | --- |
| 入参 |  |
| String mac | MAC码 |
| 返回 | 无 |

@Override

**public void** OnConnectSuccess(String mac, Connection connection)

连接成功时

|  |  |
| --- | --- |
| 入参 |  |
| String mac | MAC码 |
| Connection connection |  |
| 返回 | 无 |

@Override  
**public void** OnDiscoverService(String mac, List<BluetoothGattService> list)

发现服务时

|  |  |
| --- | --- |
| 入参 |  |
| String mac | MAC码 |
| List<BluetoothGattService> list | 发现的服务 |
| 返回 | 无 |

@Override  
**public void** OnConnectTimeout(String mac)

扫描超时时

|  |  |
| --- | --- |
| 入参 |  |
| String mac | MAC码 |
| 返回 | 无 |

@Override  
**public void** OnDisconnect(String mac, BluetoothDevice bluetoothDevice, **int** status)

断开连接时

|  |  |
| --- | --- |
| 入参 |  |
| String mac | MAC码 |
| BluetoothDevice bluetoothDevice | 蓝牙设备 |
| **int** status |  |
| 返回 | 无 |

### DisConnect

断开连接

**public void** DisConnect(**boolean** force)

|  |  |
| --- | --- |
| 入参 |  |
| **boolean** force | 默认为false |
| 返回 | 无 |

### IsConnected

获取连接状态

**public boolean** IsConnected(String mac)

|  |  |
| --- | --- |
| 入参 |  |
| String mac | MAC码 |
| 返回 | True：连接中 False：未连接 |

## 设置

### Init

初始化设备（一般用于判断设备是否在线）

**public void** Init(@NonNull IInitCallback callback)

|  |  |
| --- | --- |
| 入参 |  |
| @NonNull IInitCallback callback | 初始化回调 |
| 返回 | 无 |

* **参数说明**

**IInitCallback：**

@Override  
**public void** OnData(**int** status, **byte**[] result)

初始化成功时

|  |  |
| --- | --- |
| 入参 |  |
| **int** status | 0x00 成功 其他无效，具体定义见附录B； |
| **byte**[] result | 返回的数据 |
| 返回 | 无 |

@Override  
**public void** OnError(Throwable t)

发生错误时

|  |  |
| --- | --- |
| 入参 |  |
| Throwable t | 错误信息 |
| 返回 | 无 |

以下OnError等同此说明，故略

### GetInfo

获取版本号信息

**public void** GetInfo(@NonNull IGetInfoCallback callback)

|  |  |
| --- | --- |
| 入参 |  |
| @NonNull IGetInfoCallback callback | 获取版本号信息回调 |
| 返回 | 无 |

* **参数说明**

**IGetInfoCallback：**

@Override  
**public void** OnData(**int** status,String HardVer,String FirmVer,String SNCode, **byte**[] result)

还原出厂设置成功时

|  |  |
| --- | --- |
| 入参 |  |
| **int** status | 0x00 成功 其他无效，具体定义见附录B； |
| String HardVer | 硬件型号 |
| String FirmVer | 软件型号 |
| String SNCode | SN Code |
| **byte**[] result | 返回的数据 |
| 返回 | 无 |

### Reboot

还原出厂设置

**public void** Reboot(@NonNull IRebootCallback callback)

|  |  |
| --- | --- |
| 入参 |  |
| @NonNull IRebootCallback callback | 还原出厂设置回调 |
| 返回 | 无 |

* **参数说明**

**IRebootCallback：**

@Override  
**public void** OnData(**int** status, **byte**[] result)

还原出厂设置成功时

|  |  |
| --- | --- |
| 入参 |  |
| **int** status | 0x00 成功 其他无效，具体定义见附录B； |
| **byte**[] result | 返回的数据 |
| 返回 | 无 |

### SetPower

设置功率

**public void** SetPower(**byte** power,@NonNull ISetPowerCallback callback)

|  |  |
| --- | --- |
| 入参 |  |
| **byte** power | 功率 |
| @NonNull ISetPowerCallback callback | 功率设置回调 |
| 返回 | 无 |

* **参数说明：**

**ISetPowerCallback：**

@Override  
**public void** OnData(**int** status, **byte**[] result)

设置功率成功时

|  |  |
| --- | --- |
| 入参 |  |
| **int** status | 0x00 成功 其他无效，具体定义见附录B； |
| **byte**[] result | 返回的数据 |
| 返回 | 无 |

### GetRFID

获取RFID

**public void** GetRFID(@NonNull IGetRFIDCallback callback)

|  |  |
| --- | --- |
| 入参 |  |
| @NonNull IGetRFIDCallback callback | 获取RFID回调 |
| 返回 | 无 |

* **参数说明：**

**IGetRFIDCallback：**

@Override  
**public void** OnData(**int** status, **int** RFID, **byte**[] result)

获取RFID成功时

|  |  |
| --- | --- |
| 入参 |  |
| **int** status | 0x00 成功 其他无效，具体定义见附录B； |
| **int** RFID | 获取到的RFID |
| **byte**[] result | 返回的数据 |
| 返回 | 无 |

### SetRFID

设置RFID

**public void** SetRFID(**byte** rfid,@NonNull ISetRFIDCallback callback)

|  |  |
| --- | --- |
| 入参 |  |
| **byte** rfid | 要设置的RFID |
| @NonNull ISetRFIDCallback callback | 设置RFID回调 |
| 返回 | 无 |

* **参数说明：**

**ISetRFIDCallback：**

@Override  
**public void** OnData(**int** status, **byte**[] result)

设置RFID成功时

|  |  |
| --- | --- |
| 入参 |  |
| **int** status | 0x00 成功 其他无效，具体定义见附录B； |
| **byte**[] result | 返回的数据 |
| 返回 | 无 |

### GetBattery

获取电量

**public void** GetBattery(@NonNull IGetBatteryCallback callback)

|  |  |
| --- | --- |
| 入参 |  |
| @NonNull IGetBatteryCallback callback | 获取电量回调 |
| 返回 | 无 |

* **参数说明：**

**IGetBatteryCallback：**

@Override  
**public void** OnData(**int** status,**int** battery, **byte**[] result)

获取电量成功时

|  |  |
| --- | --- |
| 入参 |  |
| **int** status | 0x00 成功 其他无效，具体定义见附录B； |
| **int** battery | 获取到的电量 |
| **byte**[] result | 返回的数据 |
| 返回 | 无 |

### GetBLEName

获取蓝牙名称

**public void** GetBLEName(@NonNull IGetBLENameCallback callback)

|  |  |
| --- | --- |
| 入参 |  |
| @NonNull IGetBLENameCallback callback | 获取蓝牙名称回调 |
| 返回 | 无 |

* **参数说明：**

**IGetBLENameCallback：**

@Override  
**public void** OnData(**int** status,String Name, **byte**[] result)

获取蓝牙名称成功时

|  |  |
| --- | --- |
| 入参 |  |
| **int** status | 0x00 成功 其他无效，具体定义见附录B； |
| String Name | 获取到的蓝牙名称 |
| **byte**[] result | 返回的数据 |
| 返回 | 无 |

### SetBLEName

设置蓝牙名称

**public void** SetBLEName(String BLEName,@NonNull ISetBLENameCallback callback)

|  |  |
| --- | --- |
| 入参 |  |
| **byte** rfid | 要设置的蓝牙名称 |
| @NonNull ISetRFIDCallback callback | 设置蓝牙名称回调 |
| 返回 | 无 |

* **参数说明：**

**ISetRFIDCallback：**

@Override  
**public void** OnData(**int** status, **byte**[] result)

设置蓝牙名称成功时

|  |  |
| --- | --- |
| 入参 |  |
| **int** status | 0x00 成功 其他无效，具体定义见附录B； |
| **byte**[] result | 返回的数据 |
| 返回 | 无 |

### GetAllParam

获取全部参数

**public void** GetAllParam(@NonNull IGetAllParamCallback callback)

|  |  |
| --- | --- |
| 入参 |  |
| @NonNull IGetAllParamCallback callback | 获取全部参数回调 |
| 返回 | 无 |

* **参数说明：**

**IGetAllParamCallback：**

@Override  
**public void** OnData(**int** status, DeviceParam param, **byte**[] result)

获取全部参数成功时

|  |  |
| --- | --- |
| 入参 |  |
| **int** status | 0x00 成功 其他无效，具体定义见附录B； |
| DeviceParam param | 获取到的全部参数 |
| **byte**[] result | 返回的数据 |
| 返回 | 无 |

**DeviceParam：**

|  |  |
| --- | --- |
| **byte** Addr | 设备的通信地址，默认为0x00  不能为0xFF |
| **byte** RFIDPRO | 设备射频RFID的协议标准规范  0x00：ISO 18000-6C；  0x01：GB/T 29768；  0x02：GJB 7377.1；  当前仅支持 ISO 18000-6C |
| **byte** WorkMode | 设备的工作模式，默认值0  0:应答模式  1:主动模式 |
| **byte** Interface | 设备的通信接口，默认值0x80  0x01: USB  0x02: Keyboard  0x04: CDC\_COM  0x10:Wifi  0x20: RJ45  0x40:RS485  0X80:RS232  当前仅支持：RS232 |
| **byte** Baudrate | 串口波特率，默认值为4  0:9600 bps  1:19200 bps  2:38400 bps  3:57600 bps  4:115200 bps |
| **byte** WGSet | 韦根数据输出接口的配置参数，默认值0x00  Bit位注释：  Bit7—0：关闭韦根输出 1：开启韦根输出  Bit6-0：wg26 1：wg34  Bit5-0:低位在前 1：高位在前  其他bit位备用 |
| **byte** Ant | 设备所有的天线号，按位表示选择使用的天线，对应Bit位值为1则表示使用该天线，值为0则表示不使用该天线；从低位开始，第0位表示1号天线，第1位表示2号天线，以此类推，最多能表示8个天线；  默认值0x01，表示1号天线 |
| **byte** Region | 频段范围  0x01：US [902.75~927.25]  0x03：EU [865.1~868.1] |
| **byte[]** StartFreI | 兆赫兹起始频率的整数部分  长度为2  如920.125MHz  STRATFREI = 920 = 0x0398  StartFreI[0]=0x03  StartFreI[1]=0x98 |
| **byte[]** StartFreD | 兆赫兹起始频率的小数部分  长度为2  如920.125MHz  STRATFREI = 125 = 0x007D  StartFreD[0]=0x00  StartFreD[1]=0x7D |
| **byte[]** StepFre | 频率步进（KHz）  需参考各频段计算公式  如125KHz  STEPFRE =125 = 0x007D  StepFre[0]=0x00  StepFre[1]=0x7D  **频率计算公式**：  **最小频点**：Fmin = STRATFREI + STRATFRED/1000 （单位：MHz）  **最大频点**：Fmax = Fmin + STEPFRE\*CN/1000 （单位：MHz）  **各频段计算公式**：(设置的频率范围须在各国标准内)  **US band**：  Fs = 902.75 + CN \* 0.5 (MHz)  其中CN∈[0,49]。  **EU band**:  Fs = 865.1 + CN\*0.2(MHz)  其中CN∈[0, 14]。 |
| **byte** Channel | 信道数 |
| **byte** RfidPower | 设备的RFID输出功率，单位为：dBm  取值范围为：[0, 30]dBm，其他无效 |
| **byte** InquiryArea | 设备要访问标签的存储区区域  0x00：保留区  0x01(默认)：EPC存储区  0x02：TID存储区  0x03：USER存储区  0x04：EPC+TID  0x05：EPC+USER  0x06：EPC+TID+USER  其它值保留 |
| **byte** QValue | 询查EPC标签时使用的初始Q值  Q值的设置应为场内的标签数量约等于2Q  Q值的默认值为4  Q值的范围为0～15 |
| **byte** Session | 询查EPC标签时使用的Session值  默认为0  取值范围[0，3] |
| **byte** AcsAddr | 设备要访问标签存储区的起始地址  单位 ：Byte  默认值：0x00  访问EPC区时，0x00表示访问EPC区除CRC和PC段的EPC号码段起始地址;  访问其他存储区时，0x00表示该存储区的起始地址。 |
| **byte** AcsDataLen | 设备要访问标签存储区的数据长度  默认值：0x00 |
| **byte** FilterTime | 过滤时间  在读取成功一张标签数据后的该值时间内，过滤掉有相同数据的标签。  单位为：S，取值范围为：[0, 255]，其他无效  默认值为0，没有过滤 |
| **byte** TriggerTime | 设备收到触发信号后的询查时长,单位为：S  默认值为1  取值范围为：[0, 255]，其他无效 |
| **byte** BuzzerTime | 设备执行成功后蜂鸣器鸣叫时长,单位为：10ms,取值范围为：[0, 255],其他无效。  默认为1,当为0时，表示蜂鸣器不鸣叫 |
| **byte** PollingInterval | 询查间隔时间,单位为：10ms,取值范围为：[0, 255],其他无效。  默认为5 |

### SetAllParam

设置全部参数

**public void** SetAllParam(DeviceParam param,@NonNull ISetAllParamCallback callback)

|  |  |
| --- | --- |
| 入参 |  |
| DeviceParam param | 要设置的参数,详细请参照GetAllParam |
| @NonNull ISetAllParamCallback callback | 设置全部参数回调 |
| 返回 | 无 |

* **参数说明：**

**ISetRFIDCallback：**

@Override  
**public void** OnData(**int** status, **byte**[] result)

设置全部参数成功时

|  |  |
| --- | --- |
| 入参 |  |
| **int** status | 0x00 成功 其他无效，具体定义见附录B； |
| **byte**[] result | 返回的数据 |
| 返回 | 无 |

### GetBLEAddr

获取蓝牙地址

**public void** GetBLEAddr(@NonNull IGetBLEAddrCallback callback)

|  |  |
| --- | --- |
| 入参 |  |
| @NonNull IGetBLEAddrCallback callback | 获取蓝牙地址回调 |
| 返回 | 无 |

* **参数说明：**

**IGetBLEAddrCallback：**

@Override  
**public void** OnData(**int** status,String Addr, **byte**[] result)

获取蓝牙地址成功时

|  |  |
| --- | --- |
| 入参 |  |
| **int** status | 0x00 成功 其他无效，具体定义见附录B； |
| String Addr | 获取到的蓝牙地址 |
| **byte**[] result | 返回的数据 |
| 返回 | 无 |

## 盘点

### StartInventory

开始盘点

**public void** StartInventory(**byte** InvType,**int** InvParam,@NonNull IStartInventoryCallback callback)

|  |  |
| --- | --- |
| 入参 |  |
| **byte** InvType | 盘点方式  0x00：按时间盘点标签，在持续指定时间后停止盘点或接收到停止盘点命令后停止盘点；  0x01: 按照循环次数盘点，在执行指定次数的轮询后或者接到停止盘点指令后停止盘点；  0x03：自定义盘点（掩码盘点），(select 和 query 参数使用 RFM\_SET\_SELPRM 和RFM\_SET\_QUERY\_PARAM 命令设置）， 在持续指定时间后停止盘点或接收到停止盘点命令后停止盘点； |
| **int** InvParam | 盘点方式参数  1.若InvType为0x01：  InvParam表示盘点时间，单位为：秒，如果该值为0，则表示持续盘点标签，直到接收到停止盘点命令；  2.若InvType为0x02：  InvParam表示盘点次数，单位为：次，该值必须大于0；  3.若InvType为0x03：  InvParam表示盘点时间，单位为：秒，该值最大时间为255 |
| @NonNull IStartInventoryCallback callback | 开始盘点回调 |
| 返回 | 无 |

* **参数说明：**

**IStartInventoryCallback：**

@Override  
**public void** OnInventory(**int** status)

开始盘点成功时

|  |  |
| --- | --- |
| 入参 |  |
| **int** status | 0x00 成功 其他无效，具体定义见附录B； |
| 返回 | 无 |

@Override  
**public void** OnData(**int** status, InventoryTag tag,**byte**[] result)

盘点到数据时

|  |  |
| --- | --- |
| 入参 |  |
| **int** status | 0x00 成功 其他无效，具体定义见附录B； |
| InventoryTag tag |  |
| **byte**[] result | 返回的数据 |
| 返回 | 无 |

**InventoryTag：**

|  |  |
| --- | --- |
| **byte** Status | 0x00 成功 其他无效，具体定义见附录B； |
| **byte[]** RSSI | 标签ACK响应的RSSI，单位为0.1dBm，带符号数，负数使用补码格式； |
| **byte** Antenna | 天线的数目 |
| **byte** Channel | 从哪个信道接收到的标签数据，值从0开始，0表示0信道，1表示1信道，以此类推； |
| **byte** EPCLen | 标签的EPC号码长度（byte）； |
| **byte[]** EPCNum | 标签的EPC号码 |

### StopInventory

停止盘点

**public void** StopInventory(@NonNull IStopInventoryCallback callback)

|  |  |
| --- | --- |
| 入参 |  |
| @NonNull IStopInventoryCallback callback | 停止盘点回调 |
| 返回 | 无 |

* **参数说明：**

**IStopInventoryCallback：**

@Override  
**public void** OnData(**int** status,**byte**[] result)

停止盘点成功时

|  |  |
| --- | --- |
| 入参 |  |
| **int** status | 0x00 成功 其他无效，具体定义见附录B； |
| **byte**[] result | 返回的数据 |
| 返回 | 无 |

## 读写标签

### SelectTag

选择读写的标签

**public void** SelectTag(String code, @NonNull ISelectTagCallback callback)

|  |  |
| --- | --- |
| 入参 |  |
| String code |  |
| @NonNull IOperateTagCallback callback | 选择读写的标签回调 |
| 返回 | 无 |

* **参数说明：**

**IOperateTagCallback：**

@Override  
**public void** OnData(**int** status, **byte**[] result)

选择读写的标签成功时

|  |  |
| --- | --- |
| 入参 |  |
| **int** status | 0x00 成功 其他无效，具体定义见附录B； |
| **byte**[] result | 返回的数据 |
| 返回 | 无 |

### ReadTag

读标签

**public void** ReadTag(TagParam param, @NonNull IOperateTagCallback callback)

|  |  |
| --- | --- |
| 入参 |  |
| TagParam param |  |
| @NonNull IOperateTagCallback callback | 操作标签回调 |
| 返回 | 无 |

* **参数说明：**

**TagParam：**

|  |  |
| --- | --- |
| **byte**[] AccPwd | 访问口令，用于标签进入安全态，  默认0x00000000,长度为4 |
| **byte** Menbank | 所要读取标签的存储区，  值列表如下：  0x00：Reserved；  0x01：EPC；  0x02：TID；  0x03：User |
| **byte**[] WordPtr | 指向逻辑存储区的读取起始地址(字)； |
| **byte** WordCount | 需要读取的字个数；  1. WordCount等于0，MemBank 不为 01 存取区时，标签返回的数据内容是从 WordPtr 定义的起始地址到该存储区的最后。  2. WordCount等于0，MemBank 等于 01 存取区时，标签将返回的数据内容是从 WordPtr 定义的起始地址到由 PC 前 5 位所指定的 EPC 长度结束。 |

**IOperateTagCallback**:

@Override  
**public void** OnData(**int** status, OperateTag tag, **byte**[] result)

操作标签成功时

|  |  |
| --- | --- |
| 入参 |  |
| **int** status | 0x00 成功 其他无效，具体定义见附录B； |
| OperateTag tag | 操作标签返回的数据 |
| **byte**[] result | 返回的数据 |
| 返回 | 无 |

**OperateTag:**

|  |  |
| --- | --- |
| **byte** Status | 0x00 成功 其他无效，具体定义见附录B； |
| **byte** TagStatus | 标签返回的操作状态，具体定义见附录A； |
| **byte** Antenna | 从哪根天线中读出 |
| **byte**[] CRC | 标签响应数据中的CRC数据，长度为2 |
| **byte**[] PC | 标签响应数据中的PC数据，长度为2 |
| **byte** EPCLen | 标签的EPC号码长度（字节）； |
| **byte**[] EPCCode | 标签的EPC号码长度（字节）； |
| **byte** WordCount | 成功读取到的标签数据字个数；  （仅读标签的时候有数据） |
| **byte**[] Data | 成功读取到的标签数据，长度为WordCount×2个字节；  （仅读标签的时候有数据） |

### WriteTag

写标签

**public void** WriteTag(TagParam param,@NonNull IOperateTagCallback callback)

|  |  |
| --- | --- |
| 入参 |  |
| TagParam param |  |
| @NonNull IOperateTagCallback callback | 操作标签回调 |
| 返回 | 无 |

* **参数说明：**

**TagParam：**

|  |  |
| --- | --- |
| **byte**[] AccPwd | 访问口令，用于标签进入安全态，  默认0x00000000,长度为4 |
| **byte** Menbank | 所要读取标签的存储区，  值列表如下：  0x00：Reserved；  0x01：EPC；  0x02：TID；  0x03：User |
| **byte**[] WordPtr | 指向逻辑存储区的读取起始地址(字)； |
| **byte** WordCount | 需要写入标签的数据字个数(1个字为两个字节)，必须大于0； |
| **byte**[] Data | 需要写入标签的数据，长度必须为字的整数倍，长度为1 ~ WordCount个字 |

**IOperateTagCallback**:详情参见ReadTag

### LockTag

锁标签

**public void** LockTag(TagParam param,@NonNull IOperateTagCallback callback)

|  |  |
| --- | --- |
| 入参 |  |
| TagParam param |  |
| @NonNull IOperateTagCallback callback | 操作标签回调 |
| 返回 | 无 |

* **参数说明：**

**TagParam：**

|  |  |
| --- | --- |
| **byte**[] AccPwd | 访问口令，用于标签进入安全态，  默认0x00000000,长度为4 |
| **byte** Area | 需要锁定的区域，值列表如下：  0x00：灭活密码区；  0x01：访问密码区；  0x02：EPC；  0x03：TID；  0x04：User； |
| **byte** Action | 锁定操作类型，值列表如下：  0x00：开放；  0x01：永久开放；  0x02：锁定；  0x03：永久锁定； |

**IOperateTagCallback**:详情参见ReadTag

### KillTag

灭活标签

**public void** KillTag(TagParam param,@NonNull IOperateTagCallback callback)

|  |  |
| --- | --- |
| 入参 |  |
| TagParam param |  |
| @NonNull IOperateTagCallback callback | 操作标签回调 |
| 返回 | 无 |

* **参数说明：**

**TagParam：**

|  |  |
| --- | --- |
| **byte**[] KillPwd | 灭活口令，长度为4 |

**IOperateTagCallback**:详情参见ReadTag

## 附录A. 标签返回的操作状态

标签返回的操作状态编码共8位，见表格A-1。

表格A-1 标签返回的操作状态

|  |  |  |  |
| --- | --- | --- | --- |
| 操作状态编码 | 操作状态 | 描述 | 错误优先级 |
| 0x14 | 标签响应超时 |  |  |
| 0x81 | 其他错误 | 标签返回的未知错误 |  |
| 0x82 | 存储区溢出 | 目标存储区不存在 |  |
| 0x83 | 存储区锁定 | 对被锁定为不可写的存储区进行写操  作或者擦除操作，对被锁定为不可读的存储区进行读操作 |  |
| 0x84 | 功率不足 | 标签没有足够的能量完成操作 |  |
| 0x85 | 非特定错误 | 标签返回的未知错误 |  |
|  |  |  |  |

## 附录B. STATUS的定义。

表格A-6 STATUS定义

|  |  |
| --- | --- |
| STATUS | 错误描述 |
| 0x00 | 执行成功（此处只表示模块成功接收到标签响应数据，如果标签响应中有标签执行状态，则还应该进一步判断标签执行状态是否正确） |
| 0x01 | 参数值错误或越界，或者模块不支持该参数值 |
| 0x02 | 由于模块内部错误导致的命令执行失败 （设置频率或者设置功率） |
| 0x03 | 保留 |
| 0x12 | 整个盘点命令执行完成 |
| 0x13 | 没有盘点到标签 |
| 0x14 | 标签响应超时 |
| 0x15 | 解调标签响应错误 |
| 0x16 | 协议认证失败 |
| 0x17 | 口令错误 |
| 0xFF | 没有更多数据了 |