LotusCard SDK 说明文档

目 录

一、前言

凡使用 SDK 的研发人员，假定具备以下知识：

1、设备操作卡型相关知识（专业术语），如需特殊形式封装接口可以与我们联系（适当收费）用于屏蔽卡操作相关流程，如单一接口函数完成读写操作等

2、了解所使用开发语言通用知识，并读懂范例

二、版本

以下仅简单描述

1.0.0.1 R66 完成 M1 卡

1.0.0.2 R101 完成非接触式 CPU 卡（FM1208）

1.0.0.3 R117 完成多个 USB 读卡器支持,增加 WIFI 通道

1.0.0.5 R203 增加读写文本 API，支持 14443B,支持 NFC(卡模式)

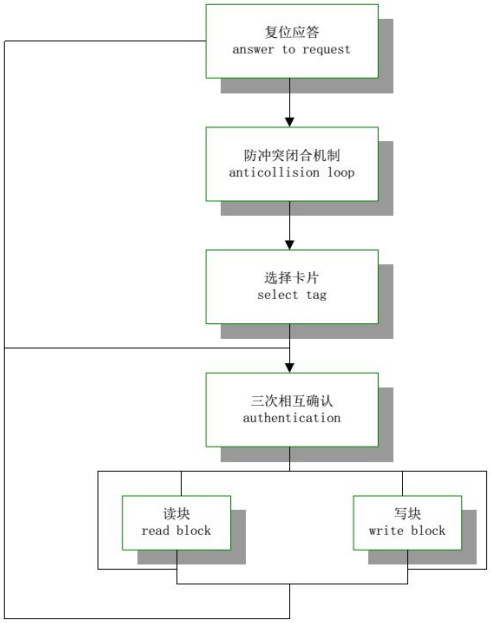
1.0.0.6 R250 增加超高频 API

三、关于 M1 卡

1. 操作流程

Request：对应调用 LotusCardRequest 函数

Anticollission：对应调用 LotusCardAnticoll 函数



Select：对应调用 LotusCardSelect 函数

Authentication: 对应调用 LotusCardAuthentication 函数

Read：对应调用 LotusCardRead 函数

Write：对应调用 LotusCardWrite 函数

2. 关于密码的处理

SDK 提供 LotusCardLoadKey 函数用于装载密码到读卡器，只能写入，不能读取，这个 API 可以单独使用，无需卡片，如果卡片密码有变化，需要调用 LotusCardLoadKey 重新装载密码。

密码装载到读卡器后，直接调用 LotusCardAuthentication 认证即可，发布到用户端的程序无需携带密码。

3. 简化流程

针对 Request、Anticollission、Select 三个动作，每次 M1 操作都会执行，SDK 提供了一个 API 封装上述 3 个动作相关函数：LotusCardGetCardNo，使用 LotusCardGetCardNo 函数即可一次性完成上述动作，简化 API 调用。

4. 相关文档

具体卡片文档详见 SDK 链接“FM11RF08 产品说明书.PDF”，“M1 卡资料(中文).doc”等文档以及相关网络资源。

四、关于 CPU 卡

设备支持 ISO14443A CPU 卡片（L3-U 支持 14443B），目前验证过 FM1208，

以及客户验证过银行卡、移动 SIM 卡（内置安全模块，NFC 天线交互）。

1、操作流程

前 3 个步骤同 M1，SDK 做了简化操作，只需直接调用复位函数 LotusCardResetCpuCard 即可。复位动作操作成功后，即可发送 COS 指令。

2、COS 指令

详见 CPU 卡相关文档说明。

五、关于 NTAG 系列卡片

设备支持 ULTRALIGHT/NTAG203/NTAG213/NTAG215/NTAG216。

1、操作流程

前 2 个步骤同 M1，无需 SELECT 以及认证动作，即可进行读写操作。当然调用简化函数 GetCardNo 后进行读写动作也是可以的，库版本在 1.0.0.6 以后支持。

2、卡片信息

详见“NXP_NTAG213_215_216_ds.pdf”。

六、关于银行卡

根据客户要求，SDK 提供了一个 API 用于获取银行卡（ISO14443A）卡号，根据 PBOC 相关标准解析，用户无需操作 COS 指令，复位成功后直接执行 LotusCardGetBankCardNo 函数即可。

七、关于动态库

1. 所有平台动态库均导出 JAVA 和标准 C 调用函数

C 调用函数增加 LotusCard 前缀

如 java 函数 OpenDevice，对应 C 函数 LotusCardOpenDevice

1、ANDROID 动态库调用说明

串口或 USB 设备需要具有读写权限

串口如果没有需要找 ANDROID 设备供应商提供解决方案

USB 如果没有可以采用 ANDROID 本身的 USB HOST 访问机制，但是需要 ANDROID3.1 以上版本

2. ActiveX 控件说明

ActiveX 控件相关方法同 DLL 定义，区别在于 DLL 使用参数结构体 LotusCardParamStruct，ActiveX 控件将参数结构体中相关定义改为了属性。

属性：CardNo 对应 LotusCardParamStruct 中 arrCardNo;

属性：ReadWriteBuffer 对应 LotusCardParamStruct 中 arrBuffer;

属性：CardPassWord 对应 LotusCardParamStruct 中 arrKeys;

3. 特殊平台需提供相关编译器

提供编译器的同时需告诉 CPU 型号，部分 CPU 存在指令限制。

动态库编译后由客户完成测试，如需我们协助，请提供相关硬件。

4. 注意事项

尽量使用范例自身携带动态库，避免版本冲突。

Android/java 由于 JNI 约定：

LotusCardDriver.java、LotusCardErrorCode.java、LotusCardParam.java

所在包不能变。

八、基础函数

函数说明以 ANDROID JAVA 定义为范例，其他开发语言类似，请参看具体范例，由于 API 持续更新中，部分函数会有参数发生变动，当然我们也尽量保持 API 稳定。

参数结构体说明：

arrCardNo 早期版本为 4 字节，用的时候要注意，arrCosResultBuffer ，arrCosSendBuffer 为新增的，如果客户需要更新动态库，LotusCardDriver.java、LotusCardErrorCode.java、LotusCardParam.java 文件同步更新。其他开发语言如果有函数定义的，更新函数定义文件，直接调用的修正参数结构定义。

```
package cc.lotuscard;
public class LotusCardParam {
 LotusCardParam()
 {
     arrCardNo = new byte[8];
     arrBuffer = new byte[64];
     arrKeys = new byte[64];
     arrCosResultBuffer =  new byte[256];
     arrCosSendBuffer = new byte[256];
 }
 /**
  * 卡片类型
  */
 public int nCardType;
 /**
  * 8 字节卡号
  */
```

```
public byte[] arrCardNo;

/**
 * 卡片容量大小
 */
public int nCardSize;

/**
 * 读写缓冲
 */
public byte[] arrBuffer;

/**
 * 缓冲大小
 */
public int nBufferSize;
/**
 * 密钥
 */
public byte[] arrKeys;

/**
 * KEYs 大小
 *
 */
public int nKeysSize;
/**
 * pCosResultBuffer COS 执行结果缓冲
 */
public byte[] arrCosResultBuffer;
```

```
    /**
     * unCosReultBufferLength COS 执行结果缓冲长度
     */
    public int unCosReultBufferLength;

    /**
     * pCosSendBuffer COS 指令发送缓冲
     */
    public byte[] arrCosSendBuffer;
    /**
     * unCosSendBufferLength COS 指令发送缓冲长度
     */
    public int unCosSendBufferLength;
   }
```

1. 打开设备：OpenDevice

参数 strDeviceName：为字符长度为 0 使用 USB 通道，否则使用串口通道

参数 nVID\nPID 参数如果为 0,动态库使用内部定义值，原则上建议使用 0，除非范例有明确赋值。

特别声明：使用外部读写接口优先，参见回调函数

```
    /**
     * 打开设备
     *
     * @param strDeviceName
     *            串口设备名称
     * @param nVID
     *            USB 设备 VID
     * @param nPID
```

```
 *            USB 设备 PID
 * @param nUsbDeviceIndex
 *            USB 设备索引
 * @param unRecvTimeOut
 *            接收超时
 * @param bUseExendReadWrite
 *            是否使用外部读写通道 如果没有设备写权限时，可以使用外部 USB 或串口进行通讯，
 *            需要改造 callBackProcess 中相关代码完成读写工作 目前范例提供 USB 操作
 * @return 设备句柄
 */
public native int OpenDevice(String strDeviceName, int nVID, int nPID,
        int nUsbDeviceIndex, int unRecvTimeOut, boolean bUseExendReadWrite);
```

2. 关闭设备：CloseDevice

设备句柄 nDeviceHandle 为 OpenDevice 返回值。

```
/**
 * 关闭设备
 *
 * @param nDeviceHandle
 *            设备句柄
 */
public native void CloseDevice(long nDeviceHandle);
```

3. 寻卡：Request

设备句柄 nDeviceHandle 为 OpenDevice 返回值。

M1/CPU/NTAG 系列卡片寻卡动作。

nRequestType 参数取值： RT_ALL = 0x52; // /< 符合 14443A 卡片

RT_NOT_HALT = 0x26; // /< 未进入休眠状态的卡

```
/**
 * 寻卡
 *
 * @param nDeviceHandle
 *            设备句柄
 * @param nRequestType
 *            请求类型
```

```
 * @param tLotusCardParam
 *            结果值 用里面的卡片类型
 * @return true = 成功
 */
public native boolean Request(long nDeviceHandle, int nRequestType,
        LotusCardParam tLotusCardParam);
```

4. 防冲突：Anticoll

设备句柄 nDeviceHandle 为 OpenDevice 返回值。

```
/**
 * 防冲突
 *
 * @param nDeviceHandle
 *            设备句柄
 * @param tLotusCardParam
 *            结果值 用里面的卡号
 * @return true = 成功
 */
public native boolean Anticoll(long nDeviceHandle,
        LotusCardParam tLotusCardParam);
```

5. 选卡：Select

设备句柄 nDeviceHandle 为 OpenDevice 返回值。

```
/**
 * 选卡
 *
 * @param nDeviceHandle
 *            设备句柄
 * @param tLotusCardParam
 *            参数(使用里面的卡号)与结果值(使用里面的卡容量大小)
 * @return true = 成功
 */
public native boolean Select(long nDeviceHandle,
        LotusCardParam tLotusCardParam);
```

6. 密钥验证：Authentication

M1 需要，CPU/NTAG 系列不需要

设备句柄 nDeviceHandle 为 OpenDevice 返回值。

nSectionIndex:S50（卡型 0x04）为 0~15，共 16 个扇区。
S70（卡型 0x02）为 0~39，共 40 个扇区。

```
/**
 * 密钥验证
 *
 * @param nDeviceHandle
 *            设备句柄
 * @param nAuthMode
 *            验证模式
 * @param nSectionIndex
 *            扇区索引
 * @param tLotusCardParam
 *            参数(使用里面的卡号)
 * @return true = 成功
 */
public native boolean Authentication(long nDeviceHandle, int nAuthMode,
        int nSectionIndex, LotusCardParam tLotusCardParam);
```

7. 卡片中止响应：Halt

中止后，卡片处于中止状态，再次寻卡如果请求类型是未中止的卡片将寻卡失败，用于处理完卡片操作后，避免重复操作应用场景。

设备句柄 nDeviceHandle 为 OpenDevice 返回值。

```
/**
 * 卡片中止响应
 *
 * @param nDeviceHandle
 *            设备句柄
 * @return true = 成功
 */
public native boolean Halt(long nDeviceHandle);
```

8. 读操作：Read

设备句柄 nDeviceHandle 为 OpenDevice 返回值。

特别说明：如果是 NTAG 系列，nAddress 参数为 PAGE 索引，每次可以读 4 个 PAGE，每个 PAGE 有 4 字节，共 16 字节。

S50：

nAddress ＝ 扇区索引（0~15） * 4 + 块（0~3）；

如读取 0 扇区 2 块，nAddress = 0*4 +2=2；

如读取 5 扇区 2 块，nAddress = 5*4 +2=22；

S70：

0~31 扇区计算方式同 S50;

```
nAddress ＝ 扇区索引（0~31） * 4 + 块（0~3）；
如读取 0 扇区 2 块，nAddress = 0*4 +2=2；
如读取 5 扇区 2 块，nAddress = 5*4 +2=22；
扇区 32~39 每个扇区 16 个块 nAddress ＝ 32 * 4 +（扇区索引-32）*16 + 块（0~15）；
如读取 32 扇区 2 块，nAddress = 32*4 +（32-32）*16+2=130；
如读取 37 扇区 12 块，nAddress = 32*4 +（37-32）*16 + 12 =220；

/**
 * 读指定地址数据
 *
 * @param nDeviceHandle
 *            设备句柄
 * @param nAddress
 *            块地址
 * @param tLotusCardParam
 *            结果值（读写缓冲）
 * @return true = 成功
 */
public native boolean Read(long nDeviceHandle, int nAddress,
        LotusCardParam tLotusCardParam);
```

9. 写操作：Write

设备句柄 nDeviceHandle 为 OpenDevice 返回值。

特别说明：如果是 NTAG 系列，nAddress 参数为 PAGE 索引，每个 PAGE 长度为 4 字节，不要写多了。M1 为块索引，每块 16 字节。

```
S50：
nAddress ＝ 扇区索引（0~15） * 4 + 块（0~3）；
如读取 0 扇区 2 块，nAddress = 0*4 +2=2；
如读取 5 扇区 2 块，nAddress = 5*4 +2=22；
S70：
0~31 扇区计算方式同 S50;
nAddress ＝ 扇区索引（0~31） * 4 + 块（0~3）；
如读取 0 扇区 2 块，nAddress = 0*4 +2=2；
如读取 5 扇区 2 块，nAddress = 5*4 +2=22；
扇区 32~39 每个扇区 16 个块 nAddress ＝ 32 * 4 +（扇区索引-32）*16 + 块（0~15）；
如读取 32 扇区 2 块，nAddress = 32*4 +（32-32）*16+2=130；
如读取 37 扇区 12 块，nAddress = 32*4 +（37-32）*16 + 12 =220；
/**
 * 写指定地址数据
 *
 * @param nDeviceHandle
 *            设备句柄
```

```
 * @param nAddress
 *            块地址
 * @param tLotusCardParam
 *            参数（读写缓冲）
 * @return true = 成功
 */
public native boolean Write(long nDeviceHandle, int nAddress,
        LotusCardParam tLotusCardParam);
```

10. 加值：Increment

设备句柄 nDeviceHandle 为 OpenDevice 返回值。

```
/**
 * 加值
 *
 * @param nDeviceHandle
 *            设备句柄
 * @param nAddress
 *            块地址
 * @param nValue
 *            值
 * @return true = 成功
 */
public native boolean Increment(long nDeviceHandle, int nAddress, int nValue);
```

11. 减值：Decreament

设备句柄 nDeviceHandle 为 OpenDevice 返回值。

```
/**
 * 减值
 *
 * @param nDeviceHandle
 *            设备句柄
 * @param nAddress
 *            块地址
 * @param nValue
 *            值
 * @return true = 成功
 */
public native boolean Decreament(long nDeviceHandle, int nAddress,
        int nValue);
```

12. 读块到寄存器：Restore

设备句柄 nDeviceHandle 为 OpenDevice 返回值。

```
/**
 * 卡数据块传入卡的内部寄存器
 *
 * @param nDeviceHandle
 *            设备句柄
 * @param nAddress
 *            块地址
 * @return true = 成功
 */
public native boolean Restore(long nDeviceHandle, int nAddress);
```

13. 寄存器写入块：Transfer

设备句柄 nDeviceHandle 为 OpenDevice 返回值。

```
/**
 * 内部寄存器传入卡的卡数据块
 *
 * @param nDeviceHandle
 *            设备句柄
 * @param nAddress
 *            块地址
 * @return true = 成功
 */
public native boolean Transfer(long nDeviceHandle, int nAddress);
```

14. 装载密钥：LoadKey

设备句柄 nDeviceHandle 为 OpenDevice 返回值。

装载密钥 A 或 B 到设备中

```
/**
 * 装载密钥
 *
 * @param nDeviceHandle
 *            设备句柄
 * @param nAuthMode
 *            验证模式
```

```
 * @param nSectionIndex
 *            扇区索引
 * @param tLotusCardParam
 *            参数（密钥）
 * @return true = 成功
 */
public native boolean LoadKey(long nDeviceHandle, int nAuthMode,
        int nSectionIndex, LotusCardParam tLotusCardParam);
```

15. 蜂鸣：Beep

设备句柄 nDeviceHandle 为 OpenDevice 返回值。

```
/**
 * 蜂鸣
 *
 * @param nDeviceHandle
 *            设备句柄
 * @param nBeepLen
 *            蜂鸣长度 毫秒为单位
 * @return true = 成功
 */
public native boolean Beep(long nDeviceHandle, int nBeepLen);
```

16. CPU 卡指令发送：SendCpuCommand

设备句柄 nDeviceHandle 为 OpenDevice 返回值。

发送 CPU 卡指令

```
/**
 * 发送指令 用于 CPU 卡
 *
 * @param nDeviceHandle
 *            设备句柄
 * @param nTimeOut
 *            超时参数
 * @param tLotusCardParam
 *            参数（指令缓冲,返回结果）
 * @return true = 成功
 */
public native boolean SendCpuCommand(long nDeviceHandle, int nTimeOut,
        LotusCardParam tLotusCardParam);
```

17. 连接测试:ConnectTest

设备句柄 nDeviceHandle 为 OpenDevice 返回值。

WIFI 专用

```
/**
 * 连接测试
 *
 * @param strServerIp
 *            服务器 IP 地址
 * @param nConnectTimeOut
 *            超时 us 为单位
 * @return true = 成功
 */
public native boolean ConnectTest(String strServerIp, int nConnectTimeOut);
```

18. 获取错误编码：GetErrorCode

设备句柄 nDeviceHandle 为 OpenDevice 返回值。

```
/**
 * 获取错误编码
 *
 * @param nDeviceHandle
 *            设备句柄
 * @return 错误编码 详见枚举值定义 LotusCardErrorCode.java
 */
public native int GetErrorCode(long nDeviceHandle);
```

19. 设置卡片类型:SetCardType

设备句柄 nDeviceHandle 为 OpenDevice 返回值。

cCardType 参数说明：

A：14443A 类型，M1 卡/NTAG/CPU 卡

B：14443B 类型，二代身份证

C：NFC 卡模式

F：Felica

```
/**
 * 设置卡片类型
 *
 * @param nDeviceHandle
 *            设备句柄
 * @param cCardType
 *            卡片类型 A='A'/'a' B='B'/'b' F='F'/'f' C='C'/'c'
```

```
 * @return true = 成功
 */
public native boolean SetCardType(long nDeviceHandle, char cCardType);
```

20. 读取 NFC 缓冲:ReadNfcBuffer

设备句柄 nDeviceHandle 为 OpenDevice 返回值。

设备为卡模式时，调用有效。

```
/**
 * 读取 NFC 缓冲
 *
 * @param nDeviceHandle
 *            设备句柄
 * @return 缓冲字串
 */
public native String ReadNfcBuffer(long nDeviceHandle);
```

21. 写入 NFC 缓冲:WriteNfcBuffer

设备句柄 nDeviceHandle 为 OpenDevice 返回值。

设备为卡模式时，调用有效。

```
/**
 * 写入 NFC 缓冲
 *
 * @param nDeviceHandle
 *            设备句柄
 * @param pszNfcBuffer
 *            缓冲地址
 * @return true = 成功
 */
public native boolean WriteNfcBuffer(long nDeviceHandle, String strNfcBuffer);
```

22. 寻 14443B 卡:RequestB

设备句柄 nDeviceHandle 为 OpenDevice 返回值。

```
/**
 * 寻卡
 *
 * @param nDeviceHandle
 *            设备句柄
 * @param nRequestType
 *            请求类型
```

```
 * @param tLotusCardParam
 *            结果值 用里面的卡片类型
 * @return true = 成功
 */
public native int RequestB(long nDeviceHandle, int nRequestType,
        LotusCardParam tLotusCardParam);
```

23. 选 14443B 卡:SelectB

设备句柄 nDeviceHandle 为 OpenDevice 返回值。

```
/**
 * 选卡
 *
 * @param nDeviceHandle
 *            设备句柄
 * @param tLotusCardParam
 *            结果值(返回数据在 arrBuffer 里面)
 * @return true = 成功
 */
public native int SelectB(long nDeviceHandle, LotusCardParam tLotusCardParam);
```

24. 中止 14443B 卡:HaltB

设备句柄 nDeviceHandle 为 OpenDevice 返回值。

```
/**
 * 卡片中止响应
 *
 * @param nDeviceHandle
 *            设备句柄
 * @return true = 成功
 */
public native int HaltB(long nDeviceHandle);
```

25. 获取设备号:GetDeviceNo

设备句柄 nDeviceHandle 为 OpenDevice 返回值。

读取唯一设备内部编号，编号在工厂生产时写入，由基于云端的生产系统维护。

```
/**
 * 获取设备号
 *
 * @param nDeviceHandle
```

```
 *            设备句柄
 * @return 读取到的字串
 */
public native String GetDeviceNo(long nDeviceHandle);
```

九、扩展函数

1. 获取卡号：GetCardNo

设备句柄 nDeviceHandle 为 OpenDevice 返回值。

```
/**
 * 获取卡号 上位机简化
 *
 * @param nDeviceHandle
 *            设备句柄
 * @param nRequestType
 *            请求类型
 * @param tLotusCardParam
 *            结果值
 * @return true = 成功
 */
public native boolean GetCardNo(long nDeviceHandle, int nRequestType,
        LotusCardParam tLotusCardParam);
```

2. 获取卡号：GetCardNoEx

设备句柄 nDeviceHandle 为 OpenDevice 返回值。

```
/**
 * 获取卡号 MCU 简化
 *
 * @param nDeviceHandle
 *            设备句柄
 * @param nRequestType
 *            请求类型
 * @param ucBeepLen
 *            蜂鸣长度 最长 255 毫秒
 * @param ucUseHalt
 *            使用中止 1=调用中止操作 0=不动作
 * @param ucUseLoop
```

```
     *              使用循环 1=读卡器内部循环获取卡号 获取到数据再返回 上位机接收超时后 应立即再次读取 0=读卡器内部只动作一次 *
     * @param tLotusCardParam
     *             结果值
     * @return true = 成功
     */
    public native boolean GetCardNoEx(long nDeviceHandle, int nRequestType,
            byte ucBeepLen, byte ucUseHalt, byte ucUseLoop,
            LotusCardParam tLotusCardParam);
```

3. 值块初始化：InitValue

设备句柄 nDeviceHandle 为 OpenDevice 返回值。

对于块数据不符合值操作要求的块需要使用这个函数，本质是调用 Write 函数，只是块数据内容是根据传入值有值操作规定生成。

Byte Number	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Description	Value				Value				Value				Adr	Adr	Adr	Adr

例如:卡片中第 6 块存储十进制的 16909060 也就是十六进制的 01020304,第 6 块中的数据应该为”04030201FBFCFDFE0403020106F906F9”

```
    /**
     * 初始值
     *
     * @param nDeviceHandle
     *             设备句柄
     * @param nAddress
     *             块地址
     * @param nValue
     *             值
     * @return true = 成功
     */
    public native boolean InitValue(long nDeviceHandle, int nAddress, int nValue);
```

4. 修改密码：ChangePassword

设备句柄 nDeviceHandle 为 OpenDevice 返回值。

ChangePassword 封装 write 动作，这个 API 会将控制信息更改，如果再次修改密码需要认证 A 密码，

再认证 B 密码，然后才能修改相应扇区密码。

```
/**
 * 修改密码 AB
 *
 * @param nDeviceHandle
 *            设备句柄
 * @param pPasswordA
 *            密码 A
 * @param pPasswordB
 *            密码 B
 * @return true = 成功
 */
public native boolean ChangePassword(long nDeviceHandle, int nSectionIndex,
        String strPasswordA, String strPasswordB);
```

5. 复位 CPU 卡：ResetCpuCard

设备句柄 nDeviceHandle 为 OpenDevice 返回值。

封装 SendCpuCommand 函数，发送复位指令。

```
/**
 * 复位 CPU 卡
 *
 * @param nDeviceHandle
 *            设备句柄
 * @param tLotusCardParam
 *            结果值
 * @return true = 成功
 */
public native boolean ResetCpuCard(long nDeviceHandle,
        LotusCardParam tLotusCardParam);
```

6. 发送 COS 指令：SendCOSCommand

设备句柄 nDeviceHandle 为 OpenDevice 返回值。

```
/**
 * 发送指令 用于 CPU 卡 封装 LotusCardSendCpuCommand
 *
 * @param nDeviceHandle
 *            设备句柄
 * @param tLotusCardParam
 *            参数（指令缓冲,返回结果）
```

```
 * @return true = 成功
 */
public native boolean SendCOSCommand(long nDeviceHandle,
        LotusCardParam tLotusCardParam);
```

7. 获取银行卡卡号:GetBankCardNo

设备句柄 nDeviceHandle 为 OpenDevice 返回值。

根据 PBOC 标准，获取银行卡号，封装 SendCOSCommand 动作。

```
/**
 * 获取银行卡卡号
 *
 * @param nDeviceHandle
 *
 * @return 银行卡卡号
 */
public native String GetBankCardNo(long nDeviceHandle);
```

8. 读指定地址数据： ReadData

设备句柄 nDeviceHandle 为 OpenDevice 返回值。

这个 API 是为了减少 WIFI 通信时交互动作而简化，其他接口也是可以用的。

```
/**
 * 读指定地址数据 一个指令就完成所有动作
 *
 * @param nDeviceHandle
 *            设备句柄
 * @param nRequestType
 *            请求类型
 * @param nAddress
 *            块地址
 * @param ucUsePassWord
 *            使用密码 1=使用参数密码 0 =使用设备内部密码
 * @param ucBeepLen
 *            蜂鸣长度 最长 255 毫秒
 * @param ucUseHalt
 *            使用中止
 * @param tLotusCardParam
 *            结果值（读写缓冲）
 * @return true = 成功
 */
public native boolean ReadData(long nDeviceHandle, int nRequestType,
```

```
        int nAddress, byte ucUseParameterPassWord, byte ucBeepLen,
        byte ucUseHalt, LotusCardParam tLotusCardParam);
```

9. 写指定地址数据:WriteData

设备句柄 nDeviceHandle 为 OpenDevice 返回值。

这个 API 是为了减少 WIFI 通信时交互动作而简化，其他接口也是可以用的。

```
/**
 * 写指定地址数据
 *
 * @param nDeviceHandle
 *            设备句柄
 * @param nAddress
 *            块地址
 * @param ucBeepLen
 *            蜂鸣长度 最长 255 毫秒
 * @param ucUseHalt
 *            使用中止
 * @param tLotusCardParam
 *            参数（读写缓冲）
 * @return true = 成功
 */
public native boolean WriteData(long nDeviceHandle, int nAddress,
        byte ucBeepLen, byte ucUseHalt, LotusCardParam tLotusCardParam);
```

10. 读指定地址文本:ReadText

设备句柄 nDeviceHandle 为 OpenDevice 返回值。

nSectionIndex 参数：M1 卡为扇区索引，NTAG 系列为 Page 索引，Ntag 系列前 4 个 PAGE 不是用户区域，参数要大于 4。

```
/**
 * 读指定地址文本
 *
 * @param nSectionIndex
 *            扇区索引
 * @return 读取到的字串
 */
public native String ReadText(long nDeviceHandle, int nSectionIndex);
```

11. 写指定地址文本:WriteText

设备句柄 nDeviceHandle 为 OpenDevice 返回值。

nSectionIndex 参数：M1 卡为扇区索引，NTAG 系列为 Page 索引，Ntag 系列前 4 个 PAGE 不是用户区域，参数要大于 4。

```
/**
 * 写指定地址文本
 *
 * @param nSectionIndex
 *            扇区索引
 * @param strTextInfo
 *            写入文本字串
 * @return true = 成功
 */
public native boolean WriteText(long nDeviceHandle, int nSectionIndex,
        String strTextInfo);
```

十、回调函数

用于外部读写接口 具体代码更改参见 ANDROID 范例