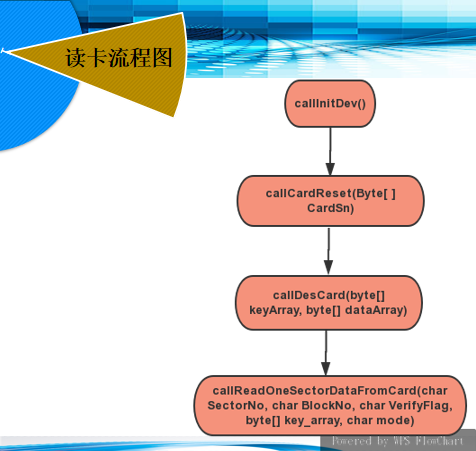
# 卡博士读写卡



|  |
| --- |
| Mifare读卡方法　byte[] callReadOneSectorDataFromCard(char SectorNo, char BlockNo, char VerifyFlag, byte[] key\_array, char mode)  返回值：这个方法　返回　byte 字节数据．此数据返回的是从Mifare 卡中读取的数据．  　　参数：SectorNo，表示的　扇区索引．  　　参数：BlockNo，表示的　域索引．  　　参数：VerifyFlag，表示的　校验标志，一般填写固定值1．  参数：key\_array，字节数据，卡片读写密钥．  　　参数：key\_array，字节数据，卡片读写方式，0x0b,写方式．0x0a读方  式． |

## Class CardLanDevCtrl

1.public int callInitDev()

init the device.

**Returns:**

int if return value equal DeviceCardConfig.INIT\_DEVICE\_STATUS\_SUCCESS, it means init devcie success, else not.

2.public int callCardReset(byte[] CardSn)

card reset

**Parameters:**

CardSn - array for saving card serial number.

**Returns:**

int if return value equal DeviceCardConfig.CARD\_RESET\_STATUS\_MONE\_SUCCESS it means mifare1 card , else equal DeviceCardConfig.CARD\_RESET\_STATUS\_CPU\_SUCCESS it means cpu card．

## 3.Class CardLanDes

public byte[] callDesCard(byte[] keyArray,

byte[] dataArray)

DES card key , the key byte array is your data source, the data byte array is your key byte array. this method only used for mifare1 card.

**Parameters:**

keyArray - source byte array

dataArray - key byte array

**Returns:**

byte[] return the des result array,the array length is DeviceCardConfig.DES\_OUT\_ARRAY\_LENGTH

## 4.Class CardLanDevCtrl

public byte[] callReadOneSectorDataFromCard(char SectorNo,

char BlockNo,

char VerifyFlag,

byte[] key\_array,

char mode)

read data from mifare1 card

**Parameters:**

SectorNo - sector index

BlockNo - area index

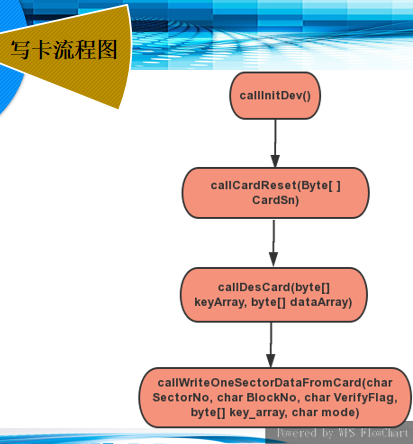
VerifyFlag - verify flag.Suggest options(1)

key\_array - key byte array

mode - key check mode. suggest options(0x0a(read)/0x0b(write)).

**Returns:**

byte[] return byte array data



|  |
| --- |
| Mifare　卡写方法　int callWriteOneSertorDataToCard(byte[] SectorArray, char SectorNo, char　BlockNo, char VerifyFlag, byte[] key\_array, char mode)  返回值：int值. 通过此值可以判断是否写卡成功．  　　参数：SectorArray，字节数组，表示需要写入的扇区域数据  　　参数：SectorNo，表示的　扇区索引．  　　参数：BlockNo，表示的　域索引．  　　参数：VerifyFlag，表示的　校验标志，一般填写固定值1．  参数：key\_array，字节数据，卡片读写密钥．  参数：key\_array，字节数据，卡片读写方式，0x0b,写方式．0x0a读 方式． |

Class CardLanDevCtrl

public int callWriteOneSertorDataToCard(byte[] SectorArray,

char SectorNo,

char BlockNo,

char VerifyFlag,

byte[] key\_array,

char mode)

write data array into mifare1 card

**Parameters:**

SectorArray - write data source array

SectorNo - sector index

BlockNo - area index

VerifyFlag - verify flag.Suggest options(1)

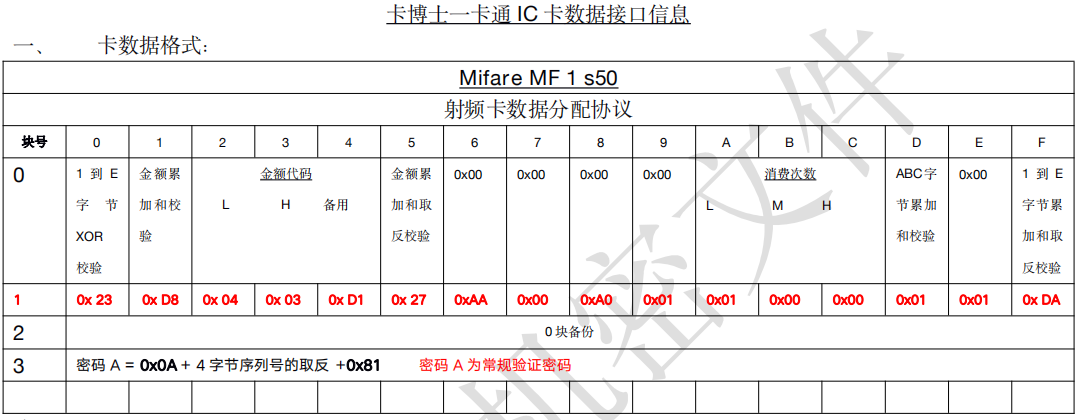
key\_array - key byte array

mode - key check mode. suggest options(0x0a(read)/0x0b(write)).

**Returns:**

int return status, if the value equal DeviceCardConfig.MONE\_CARD\_WRITE\_SUCCESS\_STATUS , it means write data success.





注：

1、IC 卡数据为十六进制；

2、金额精确到分，即以分存储数据；

3、金 额 代 码 ， 如 100.00 元 ， 其 十 六 进 制 数 为 2710 ， 则 其 金 额 代 码 为 102700 ， 金 额 累 加 和 校 验 值

=0x10+0x27+0x00=0x37；金额累加和取反校验值= 0xC8；

4、限额低位、限额高位和限额日/月未使用，其值为 0x00；

5、消费次数，办卡时，消费次数设为 1，然后每对卡金额操作一次则消费次数加 1，如新卡其消费次数为 1，即

0x010000，消费次数要按低中高位存储；

6、更新块 0 时，同时要更新块 2，块 2 数据为块 0 数据的备份数据；

7、转款只需要改写 2 字节和 3 字节的金额块即可，金额范围 0-65535（16 进制最大 655.35 元）。4 字节、6-E 字

节，不能改变其内容。改写金额完毕后，除改变金额校验外，整块的校验 0 字节和 F 字节也需要改变。遵循先

写块 2 再写块 0 的原则，块 2 是块 0 的备份块。

二、

卡读写密钥：

KEYA = 0x0A + 4 字节序列号的取反 +0x81

密码 A 为常规验证密码

控制字=7F078869

KEYB = A2020BC1027D

三、

使用扇区：



卡机所使识别的扇区号为 10 扇区号，扇区号由 0 开始；

注：

卡序列号说明，如卡片 0 扇区 0 块数据如下：

97 71 89 2A 45 08 04 00 01 1C FB 39 6D 5A E2 1D

则卡序列号为 97 71 89 2A，则卡序列号取反的值为 688E76D5。



2020-11-20