Name	**CLOUDPOS SDK 开发文档**		
Version	**1.3**	**Number**	
Date	01/18/2016	**State**	[ ]Draft [√]Release [ ]Revising

CLOUDPOS SDK 开发文档

Prepared by		**Approved By**	
Date		**Date**	

Version History

Version	Author	Date	Description
1.0	Jack.zhang	06/20/2015	Document creation
1.1	Maggie.ma	12/16/2015	排版优化，添加键盘物理按键的说明，添加 APN 设置接口说明。
1.3	Maggie.ma	01/18/2016	添加指纹模块说明

目录

1 概述 6

1.1 目的 6

1.2 规范性引用文件 6

1.3 术语和定义 6

1.3.1 银行卡 Bank Card 6

1.3.2 持卡人 Card Holder 7

1.3.3 磁条卡 MSR 7

1.3.4 集成电路（IC）卡 Integrated Circuit Card 7

1.3.5 个人标识码 Personal Identification Number (PIN) 7

1.3.6 敏感数据（信息）Sensitive Data (Information) 7

1.3.7 API 7

2 API 及使用说明 7

2.1 磁条卡阅读器 7

2.1.1 驱动文件 7

2.1.2 API 接口 8

2.1.3 硬件错误码 13

2.1.4 调用顺序 13

2.2 接触式 IC 卡阅读器 13

2.2.1 驱动文件 13

2.2.2 API 接口 14

2.2.3 硬件错误码 26

2.2.4 调用顺序 26

2.3 非接触式 IC 卡阅读器 26

2.3.1 驱动文件 26

2.3.2 API 接口 26

2.3.3 硬件错误码 40

2.3.4 调用顺序 40

2.4 密码键盘 40

2.4.1 驱动文件 40

2.4.2 API 接口 41

2.4.3 硬件错误码 53

2.4.4 调用顺序 54

2.5 打印机 54

2.5.1 驱动文件 54

2.5.2 API 接口 54

2.5.3 硬件错误码 58
2.5.4 调用顺序 58
2.6 LED 59
2.6.1 驱动文件 59
2.6.2 API 接口 59
2.6.3 硬件错误码 62
2.6.4 调用顺序 62
2.7 串口 62
2.7.1 驱动文件 62
2.7.2 API 接口 63
2.7.3 硬件错误码 67
2.7.4 调用顺序 67
2.8 客显 68
2.8.1 驱动文件 68
2.8.2 API 接口 68
2.8.3 硬件错误码 71
2.8.4 调用顺序 71
2.9 身份证阅读器 71
2.9.1 驱动文件 71
2.9.2 API 接口 72
2.9.3 硬件错误码 75
2.9.4 调用顺序 75
2.10 钱箱 75
2.10.1 驱动文件 75
2.10.2 API 接口 75
2.10.3 硬件错误码 78
2.10.4 调用顺序 78
2.11 键盘 78
2.11.1 键盘特殊键值 78
2.11.2 扫描键和二维码键 79
2.11.3 用法说明 79
2.11.4 使用范例 79
2.12 指纹 80
2.12.1 驱动文件 80
2.12.2 API 接口 80
2.13 手机上网 APN 配置 85
2.13.1 API 接口 85

2.13.2 用法说明 88
2.14 参数下载完成处理 88
2.14.1 流程 88
2.15 W1 上屏蔽 home 键 90
2.15.1 指定整个 apk 屏蔽/捕获 home 键 90
2.15.2 指定某个 activity 捕获 home 键 90
2.16 Q1 上的全屏处理 91
2.16.1 使用 Android 方式 91
2.16.2 使用系统接口 91
2.17 访问权限 92
2.17.1 权限位置 92
2.17.2 权限定义 92
2.18 错误码 93
2.18.1 简介 93

1 概述

1.1目的

本文档对应用软件的系统功能、应用管理等做具体规范。规定了通过应用接入网络的智能 pos 设备的应用管理规范。

本文档适用于商户收单使用的支付终端，不适用于持卡人本人使用的支付终端。

1.2规范性引用文件

下列文件对于本文件的应用是必不可少的。凡是注日期的引用文件，仅所注日期的版本适用于本文件。凡是不注日期的引用文件，其最新版本（包括所有的修改单）适用于本文件。

GB/T 2312-1980 信息交换用汉字编码字符集基本集
GB/T 4943.1-2011 信息技术设备的安全
GB/T 6833.2～6833.6-1987 电子测试仪器的电磁兼容性试验规范
GB/T 9254-2008 信息技术设备的无线电干扰极限值和测试方法
GB/T 14916-1994 识别卡物理特性
GB/T 15120.1-.5-1994 识别卡 记录技术
GB/T 15694.1-1995 识别卡 发卡者标识编号体系
GB/T 17552-1998 识别卡 金融交易卡
JR/T 0008-2000 银行卡发卡行标识代码及卡号（2001-01-01 实施）
JR/T 0025-2013 中国金融集成电路(IC)卡规范（PBOC 3.0）
Q/CUP 019-2010 非接触式读写器接口规范
Q/CUP 007 银联卡受理终端安全规范
ISO 7812-2-1993 识别卡 发卡方的标识
ISO 7816 识别卡 带触点的集成电路卡
ANSI X9.8 银行业 个人标识码的管理和安全

1.3术语和定义

1.3.1 银行卡 Bank Card

商业银行等金融机构及邮政储汇机构向社会发行的，具有消费信用、转账结算、存取现金等全部或部分功能的信用支付工具。

1.3.2 持卡人 Card Holder

银行卡的合法持有人，即与卡对应的银行账户相联系的客户。

1.3.3 磁条卡 MSR

物理特性符合 GB/T 14916 标准，磁条记录符合 GB/T 15120、GB/T 15694-1、ISO 7812-2、GB/T17552 标准的银行卡片。

1.3.4 集成电路（IC）卡 Integrated Circuit Card

内部封装一个或多个集成电路用于执行处理和存储功能的卡片。

1.3.5 个人标识码 Personal Identification Number (PIN)

即个人密码，是在联机交易中识别持卡人身份合法性的数据信息，在计算机和网络系统中任何环节都不允许 PIN 以明文的方式出现。

1.3.6 敏感数据（信息）Sensitive Data (Information)

是指磁道信息、PIN 和加密密钥等终端或持卡人独有的数据或信息，敏感数据进行有效保护，防止泄露、被修改或被破坏。

1.3.7 API

API 即应用程序的调用接口。API 是操作系统留给应用程序的一个调用接口。应用程序通过调用操作系统的 API 而使操作系统去执行应用程序的命令。

2 API 及使用说明

2.1磁条卡阅读器

2.1.1 驱动文件

驱动文件名为“libUnionpayCloudPos.so”。

对应的 C 接口文件名为“msr_driver_interface.h”

2.1.2 API 接口

open

(1) 功能描述

打开磁条卡阅读器设备。如果成功打开，则设备对象与磁条卡阅读器之间将建立连接，使得该对象能够使用下面的操作。

(2) 接口格式

int msr_open()

(3) 参数说明

无参数。

(4) 返回值

返回值：　　>=0，成功;
　　<0，错误码。

(5) 错误码定义

参见 错误码 and 硬件错误码。

(6) 使用说明

该操作应该在其他操作之前调用。

close

(1) 功能描述

关闭已经打开的磁条卡读卡器设备。如果想要调用其他接口，需要再次 open 该设备。

(2) 接口格式

int msr_close ()

(3) 参数说明

无参数。

(4) 返回值

返回值： >=0，成功;
<0，错误码。

(5) 错误码定义

参见 错误码 and 硬件错误码。

(6) 使用说明

Open 和 close 是成对的操作。当你不想再使用该设备的时候，请调用 close 来释放该设备。

register_notifier

(1) 功能描述

注册监听器。当数据准备好的时候，回调就会被调用。

(2) 接口格式

int msr_register_notifier(MSR_NOTIFIER pNotifier, void* pUserData)

(3) 参数说明

参数	类型	说明
pNotifier	MSR_NOTIFIER	磁条卡读卡器的回调
pUserData	void*	用户数据

(4) 返回值

返回值： >=0，成功;

<0，错误码。

(5) 错误码定义

参见 错误码 and 硬件错误码。

(6) 使用说明

typedef void (*MSR_NOTIFIER)(void* pUserData)。
在打开设备后，如果你想要监听刷卡信息，则需要调用该 api。

unregister_notifier

(1) 功能描述

注销监听器。

(2) 接口格式

int msr_unregister_notifier()

(3) 参数说明

无参数。

(4) 返回值

返回值： >=0，成功;
<0，错误码。

(5) 错误码定义

参见 错误码 and 硬件错误码。

(6) 使用说明

如果你已经注册了监听器且不再监听刷卡，则调用该 api 来注销该监听器。

get_track_error

(1) 功能描述

从指定的磁道获取错误信息。

(2) 接口格式

int msr_get_track_error(int nTrackIndex)

(3) 参数说明

参数	类型	说明
nTrackIndex	int	磁道索引

(4) 返回值

返回值： >=0, no error;
<0，错误码。

(5) 错误码定义

参见 错误码 and 硬件错误码。

(6) 使用说明

磁道索引取值为 0,1,2。
当监听器发现一张卡后，如果你想从该卡的指定磁道中获得信息，你应该首先调用此 api。只有当返回值>=0，下面的接口才能被调用。

get_track_data_length

(1) 功能描述

获取指定磁道的信息长度。

(2) 接口格式

int msr_get_track_data_length(int nTrackIndex)

(3) 参数说明

参数	类型	说明
nTrackIndex	int	磁道索引

(4) 返回值

返回值：　　>=0, 磁道数据的长度;
<0，错误码。

(5) 错误码定义

参见 错误码 and 硬件错误码。

(6) 使用说明

你可以使用获取的信息长度为指定磁道的信息分配内存空间

get_track_data

(1) 功能描述

从指定的磁道中获取数据信息。

(2) 接口格式

int msr_get_track_data (int nTrackIndex, unsigned char* pTrackData, int nLength)

(3) 参数说明

参数	类型	说明
nTrackIndex	int	磁道索引
pTrackData	unsigned char*	数据缓冲区
nLength	int	缓冲区长度

(4) 返回值

返回值：　　>=0, length of track data;
<0，错误码。

(5) 错误码定义

参见 错误码 and 硬件错误码。

2.1.3 硬件错误码

错误名	取值	说明
SUCCESS	0x00	success
GENERAL	0x01	General error
RESP_STX	0x40	Mismatch in the field of STX
RESP_CLASS	0x41	Mismatch in the field of class
RESP_FUNCTION	0x42	Mismatch in the field of function
RESP_ LENGTH	0x43	Mismatch in the field of length
RESP_ ETX	0x44	Mismatch in the field of ETX
RESP_ LRC	0x45	Mismatch in the field of LRC
RESP_ TRACK_MODE	0x46	Mismatch in the field of MODE
DATA_PREAMBLE	0x51	Preamble error in card read data
DATA_ POSTAMBLE	0x52	Postamble error in card read data
DATA_ LRC	0x53	LRC error in card read data
DATA_ PARITY	0x54	Parity error in card read data
DATA_ BLANK_TRACK	0x55	Blank track
CMD_STX_ETX	0x61	STX/ETX error in command communication
CMD_CLASS_FUNC	0x62	Class/Function un-recognizable in command
CMD_BCC	0x63	BCC error in command communication
CMD_LENGTH	0x64	Length error in command communication
CMD_NO_DATA	0x65	No data available to re-read
OPT_NO_MORE_SPACE	0x71	No more space available for OPT write
OPT_WRITE_TRY_WITHOUT_DATA	0x72	OTP write try without data
OPT_CRC_ERROR_IN_READ_DATA	0x73	CRC error in read data from OTP
OPT_NO_DATA	0x74	No data stored in OTP

2.1.4 调用顺序

open→register_notifier→get_track_error→get_track_data_length→get_track_data→ unregister_notifier→close

2.2接触式 IC 卡阅读器

2.2.1 驱动文件

驱动文件名为“libUnionpayCloudPos.so”。
对应的 C 接口文件名为“smart_card_interface.h”

2.2.2 API 接口

query_max_number

(1) 功能描述

查询 IC 卡阅读器的最大卡槽数。

(2) 接口格式

int smart_card_query_max_number()

(3) 参数说明

无参数。

(4) 返回值

返回值：　　> 0，卡槽数目；
==0，未定义；
<0，错误码。

(5) 错误码定义

参见 错误码 and 硬件错误码。

(6) 使用说明

IC 卡的卡槽索引一般为 0，其他为 PSAM 卡卡槽。

query_presence

(1) 功能描述

查询指定的卡槽中是否有卡。

(2) 接口格式

int smart_card_query_presence(int nSlotIndex)

(3) 参数说明

参数	类型	说明
nSlotIndex	int	卡槽索引，取值为 0 到 MAX-1

(4) 返回值

返回值：　　> 0，有卡；
==0，没有卡；
<0，错误码。

(5) 错误码定义

参见 错误码 and 硬件错误码。

(6) 使用说明

该功能可以在 open 之前调用。

open

(1) 功能描述

打开指定卡槽的接触式 IC 卡阅读器。

(2) 接口格式

int smart_card_open(int nSlotIndex, SMART_CARD_NOTIFIER pNotify, void* pUserData)

(3) 参数说明

参数	类型	说明
nSlotIndex	int	卡槽索引，取值为 0 到 MAX-1
pNotify	SMART_CARD_NOTIFIER	IC 卡监听器
pUserData	void*	用户数据

(4) 返回值

返回值：　　>= 0，成功，值为设备的句柄；
<0，错误码。

(5) 错误码定义

参见 错误码 and 硬件错误码。

(6) 使用说明

该 api 注册了监听器。

typedef void (*SMART_CARD_NOTIFIER)(void* pUserData, int nSlotIndex, int nEvent)

nEvent:
#define SMART_CARD_EVENT_INSERT_CARD 0
#define SMART_CARD_EVENT_REMOVE_CARD 1
#define SMART_CARD_EVENT_POWER_ON 2
#define SMART_CARD_EVENT_POWER_OFF 3

close

(1) 功能描述

关闭已经打开的指定卡槽的 IC 卡阅读器

(2) 接口格式

int smart_card_close(int nHandle)

(3) 参数说明

参数	类型	说明
nHandle	int	设备句柄，open 的返回值

(4) 返回值

返回值：
>= 0，成功;
<0，错误码。

(5) 错误码定义

参见 错误码 and 硬件错误码。

(6) 使用说明

Open 和 close 是成对的操作。当你不想再使用该设备的时候，请调用 close 来释放该设备。

打开和关闭设备的参数 nSlotIndex 应该一致

set_slot_info

(1) 功能描述

本方法用来向存储卡写数据。

(2) 接口格式

int smart_card_set_slot_info(int nHandle, SMART_CARD_SLOT_INFO* pSlotInfo)

(3) 参数说明

参数	类型	说明
nHandle	int	设备句柄，open 的返回值
pSlotInfo	SMART_CARD_SLOT_INFO*	卡槽信息

(4) 返回值

返回值：　　>= 0，成功;
　　<0，错误码。

(5) 错误码定义

参见 错误码 and 硬件错误码。

(6) 使用说明

```
typedef struct smart_card_slot_info
{
    /*参见标准 ISO 7816-3 中关于 TA1 的说明 */
    unsigned char FIDI;
    /*额外警戒时间（ Extra Guard Time） (ISO 7816-3 标准中参数 TC1 或 N) */
    unsigned char EGT;
    /*该参数的定义和使用协议相关，如果协议为 T=0, 该参数的含义为等待时间（Waiting Integer） (缺省值是 10，参见 ISO 7816-3 标准中关于 TC2 的说明)
    *如果协议为 T=1，该参数的含义为块和字符等待时间（Block and Character Waiting Time Integer）。参见 ISO 7816-3 标准中关于 TB3 的说明
     */
    unsigned char WI;
    /*如果协议是 T=1，该参数表示等待时间延长（Waiting Time Extention）*/
    unsigned char WTX;
```

```
    /*如果协议为 T=1，该参数表示校验计算方式（mode of EDC）
     *HAL_SCS_EDC_LRC 或者 HAL_SCS_EDC_CRC (缺省为 LRC) */
    unsigned char EDC;
   /* 协议类型，可以是：
     *   HAL_SCS_PROTOCOL_T0 ：用于 CPU 卡
     *   HAL_SCS_PROTOCOL_T1 ：用于 CPU 卡
     *   HAL_SCS_PROTOCOL_RAW：用于 memory 类卡
   */
    unsigned char protocol;
    /* 供电电压：
     *   HAL_SCS_POWER_1_8V
     *   HAL_SCS_POWER_3V
     *   HAL_SCS_POWER_5V
     */
    unsigned char power;
    /*传输字节的方式
     *   HAL_SCS_CONV_DIRECT
     *   HAL_SCS_CONV_INVERSE */
    unsigned char conv;
   /* 如果协议是 T=1，该参数表示卡的字段大小（Field Size for the Card），可以参
看 ISO 7816-3 标准中关于 TA3 的说明
   */
    unsigned char IFSC;
    unsigned char   reserved[3];
    /*可以用来设置字符等待时间（Character Waiting Time） */
    unsigned int cwt;
    /*可以用来设置块等待时间（Block Waiting Time） */
    unsigned int bwt;
    /*   nSlotInfoItem ：由以下常量通过或（OR）操作构成，用来指定目前结构中哪
     些位得到了更新，或哪些项需要设置。
              SMART_CARD_SLOT_INFO_FIDI
              SMART_CARD_SLOT_INFO_EGT
              SMART_CARD_SLOT_INFO_WI
              SMART_CARD_SLOT_INFO_WTX
              SMART_CARD_SLOT_INFO_EDC
              SMART_CARD_SLOT_INFO_PROTOCOL
              SMART_CARD_SLOT_INFO_POWER
              SMART_CARD_SLOT_INFO_CONV
              SMART_CARD_SLOT_INFO_IFSC
              SMART_CARD_SLOT_INFO_CWT
              SMART_CARD_SLOT_INFO_BWT
     */
    unsigned int nSlotInfoItem;
}SMART_CARD_SLOT_INFO;
```

power_on

(1) 功能描述

为卡槽中的 IC 卡上电。

(2) 接口格式

int smart_card_power_on(int nHandle, unsigned char* pATR, unsigned int* pATRLength, SMART_CARD_SLOT_INFO* pSlotInfo)

(3) 参数说明

参数	类型	说明
nHandle	int	设备句柄，open 的返回值
pATR	unsigned char*	[out]IC 卡的 ATR 信息
pATRLength	unsigned int*	[out]ATR 的长度
pSlotInfo	SMART_CARD_SLOT_INFO*	卡槽信息

(4) 返回值

返回值：　　> 0，成功;
　　<=0，错误码。

(5) 错误码定义

参见 错误码 and 硬件错误码。

power_off

(1) 功能描述

为卡槽中的卡下电。

(2) 接口格式

int smart_card_power_off(int nHandle)

(3) 参数说明

参数	类型	说明
nHandle	int	设备句柄，open 的返回值

(4) 返回值

返回值：　　>= 0，成功;
　　<0，错误码。

(5) 错误码定义

参见 错误码 and 硬件错误码。

(6) 使用说明

该 api 应在 power_on 后调用。
power_on 和 power_off 是成对的操作。

transmit

(1) 功能描述

该方法向 IC 卡发送 APDU 命令，并获得卡对 APDU 命令的答复，以及增加 SW1 和 SW2 字段的状态值。

(2) 接口格式

int smart_card_transmit(int nHandle, unsigned char* pAPDU, unsigned int nAPDULength, unsigned char* pResponse, unsigned int *pResponseLength)

(3) 参数说明

参数	类型	说明
nHandle	int	设备句柄，open 的返回值
pAPDU	unsigned char*	ADPU 命令
nAPDULength	unsigned int	命令长度
pResponse	unsigned char*	卡对 APDU 命令的答复
pResponseLength	unsigned int	答复的长度

(4) 返回值

返回值： >= 0，成功;
<0，错误码。

(5) 错误码定义

参见 错误码 and 硬件错误码。

(6) 使用说明

该 api 应在 power_on 和 power_off 之间调用。
如果卡槽中插入的是 CPU 卡，则在 open 以后调用 power_on，transmit 和 power_off。

mc_verify

(1) 功能描述

如果插入的卡是存储卡，则验证卡的密码。

(2) 接口格式

int smart_card_mc_verify_data(int nHandle, unsigned char* pData, unsigned int nDataLength)

(3) 参数说明

参数	类型	说明
nHandle	int	设备句柄，open 的返回值
pData	unsigned char*	密钥
nDataLength	unsigned int	密钥长度

(4) 返回值

返回值： >= 0，成功;
<0，错误码。

(5) 错误码定义

参见 错误码 and 硬件错误码。

mc_read

(1) 功能描述

读取存储卡的信息。

(2) 接口格式

int smart_card_mc_read(int nHandle, unsigned int nAreaType, unsigned char* pDataBuffer, unsigned int nDataLength, unsigned char cStartAddress)

(3) 参数说明

参数	类型	说明
nHandle	int	设备句柄，open 的返回值
nAreaType	unsigned int	Area 类型: 0--main memory, 1--protected memory 2--security momory
pDataBuffer	unsigned char*	数据缓冲区
nDataLength	unsigned int	期望读取的数据长度
cStartAddress	unsigned char	开始读取的位置

(4) 返回值

返回值： >= 0，成功，值为读取的数据长度;
<0，错误码。

(5) 错误码定义

参见 错误码 and 硬件错误码。

mc_write

(1) 功能描述

向存储卡写入数据。

(2) 接口格式

int smart_card_mc_write(int nHandle, unsigned int nAreaType, unsigned char* pData, unsigned int nDataLength, unsigned char cStartAddress)

(3) 参数说明

参数	类型	说明
nHandle	int	设备句柄，open 的返回值
nAreaType	unsigned int	Area 类型: 0--main memory, 1--protected memory 2--security momory
pData	unsigned char*	数据缓冲区
nDataLength	unsigned int	待写入的数据长度
cStartAddress	unsigned char	开始写入的位置

(4) 返回值

返回值： >= 0，成功，值为写入的数据长度;
<0，错误码。

(5) 错误码定义

参见 错误码 and 硬件错误码。

(6) 使用说明

mc_read 和 mc_write api 应该在 mc_verify 后调用。
若卡槽中插入的卡是存储卡，则在 open 以后调用 mc_verify，mc_read 和 mc_write。

mc_verify_E

(1) 功能描述

验证卡的密码。如果卡是 at88sc102 卡， 那么必须使用这个接口校验。

(2) 接口格式

int smart_card_mc_verify_data_E (int nHandle, unsigned char* pData, unsigned int nDataLength, unsigned int nAddress)

(3) 参数说明

参数	类型	说明
nHandle	int	设备句柄，open 的返回值
pData	unsigned char*	密钥

nDataLength	unsigned int	密钥长度
nAddress	unsigned int	地址：80--用户密码地址, 688--擦除应用区 1 时的 key 所在的地址 1248--擦除应用区 2 时的 key 所在的地址

(4) 返回值

返回值： >= 0，成功;
<0，错误码。

(5) 错误码定义

参见 错误码 and 硬件错误码。

mc_read_E

(1) 功能描述

读取存储卡的信息。如果卡是 at88sc102 卡， 那么必须使用这个接口读取。

(2) 接口格式

int smart_card_mc_read_E (int nHandle, unsigned int nAreaType, unsigned char* pDataBuffer, unsigned int nDataLength, unsigned char cStartAddress)

(3) 参数说明

参数	类型	说明
nHandle	int	设备句柄，open 的返回值
nAreaType	unsigned int	Area 类型: 0--main memory, 1--protected memory 2--security momory
pDataBuffer	unsigned char*	数据缓冲区
nDataLength	unsigned int	期望读取的数据长度
cStartAddress	unsigned char	开始读取的位置

(4) 返回值

返回值： >= 0，成功，值为读取的数据长度;
<0，错误码。

(5) 错误码定义

参见 错误码 and 硬件错误码。

mc_write_E

(1) 功能描述

向存储卡写入数据。如果卡是 at88sc102 卡， 那么必须使用这个接口读取。

(2) 接口格式

int smart_card_mc_write_E (int nHandle, unsigned int nAreaType, unsigned char* pData, unsigned int nDataLength, unsigned char cStartAddress)

(3) 参数说明

参数	类型	说明
nHandle	int	设备句柄，open 的返回值
nAreaType	unsigned int	Area 类型: 0--main memory, 1--protected memory 2--security momory
pData	unsigned char*	数据缓冲区
nDataLength	unsigned int	待写入的数据长度
cStartAddress	unsigned char	开始写入的位置

(4) 返回值

返回值： >= 0，成功，值为写入的数据长度;
<0，错误码。

(5) 错误码定义

参见 错误码 and 硬件错误码。

(6) 使用说明

mc_read_E 和 mc_write_E api 应该在 mc_verify_E 后调用。

2.2.3 硬件错误码

错误名	取值	说明
CMD_NO_ERROR	0x00	success
CMD_FAILED	0x01	CMD_FAILED
MSG_TYPE_NOT_MATCH	0x10	AU9540_MSG_TYPE_NOT_MATCH
MSG_SLOT_NOT_MATCH	0x11	AU9540_MSG_SLOT_NOT_MATCH
MSG_SEQ_NOT_MATCH	0x12	AU9540_MSG_SEQ_NOT_MATCH
MSG_NEED_MORE_WAIT_TIME	0x13	AU9540_MSG_NEED_MORE_WAIT_TIME
PROCEDURE_BYTE_CONFLICT	0xF4	PROCEDURE_BYTE_CONFLICT
ICC_PROTOCOL_NOT_SUPPORTED	0xF6	ICC_PROTOCOL_NOT_SUPPORTED
BAD_ATR_TCK	0xF7	BAD_ATR_TCK
BAD_ATR_TS	0xF8	BAD_ATR_TS
HW_ERROR	0xFB	An all inclusive hardware error occurred
XFR_PARITY_ERROR	0xFD	Parity error while talking to the ICC
ICC_MUTE	0xFE	timed out while talking to the ICC
CMD_ABORTED	0xFF	Host aborted the current activity

2.2.4 调用顺序

query_max_number 和 query_presence 可单独调用，不依赖 open 和 close。

如果插入的卡是 CPU 卡，正确的调用顺序为：

open→power_on→transmit→power_off→ close

如果插入的卡是存储卡，正确的调用顺序为：

open→ power_on → mc_verify→mc_read/mc_write→ power_off → close

2.3非接触式 IC 卡阅读器

2.3.1 驱动文件

驱动文件名为“libUnionpayCloudPos.so”。

对应的 C 接口文件名为“contactless_card_interface.h”

2.3.2 API 接口

open

(1) 功能描述

初始化非接触式 IC 卡阅读器。

(2) 接口格式

void* contactless_card_open(CONTACTLESS_CARD_NOTIFIER fNotifier, void* pUserData, int* pErrorCode)

(3) 参数说明

参数	类型	说明
fNotifier	CONTACTLESS_CARD_NOTIFIER	监听器
pUserData	void*	用户数据
pErrorCode	int*	[out]错误码，如果返回值为 0

(4) 返回值

返回值： 其它，成功，值为设备句柄;
0，错误码。

(5) 错误码定义

参见 错误码 and 硬件错误码。

(6) 使用说明

open 一般是和 search_target_begin 一起调用的。

search_target_begin

(1) 功能描述

开始寻卡。如果设置 nCardMode 为 auto，则阅读器会搜寻所有类型的卡；如果 nCardMode 设为 type A, type B, 或 type C，阅读器只会搜寻指定类型的卡。

(2) 接口格式

int contactless_card_search_target_begin(int nHandle, int nCardMode, int nFlagSearchAll, int nTimeout_MS)

(3) 参数说明

参数	类型	说明

nHandle	int	设备句柄，open 的返回值
nCardMode	int	寻卡模式
nFlagSearchAll	int	未使用
nTimeout_MS	int	毫秒计数的超时。若小于 0，则会一直等待。

(4) 返回值

返回值：　　>= 0，成功;
　　<0，错误码。

(5) 错误码定义

参见 错误码 and 硬件错误码。

(6) 使用说明

nCardMode 取值 ：

```
#define CONTACTLESS_CARD_MODE_AUTO      0
#define CONTACTLESS_CARD_MODE_TYPE_A    1
#define CONTACTLESS_CARD_MODE_TYPE_B    2
#define CONTACTLESS_CARD_MODE_TYPE_C    3
```

可以使用 search_target_end 来终止该方法。

search_target_end

(1) 功能描述

停止寻卡活动。

(2) 接口格式

int contactless_card_search_target_end(int nHandle)

(3) 参数说明

参数	类型	说明
nHandle	int	设备句柄，open 的返回值

(4) 返回值

返回值：　　>= 0，成功;
　　<0，错误码。

(5) 错误码定义

参见 错误码 and 硬件错误码。

(6) 使用说明

search_target_begin 和 search_target_end 是成对的操作。

close

(1) 功能描述

关闭非接卡阅读器

(2) 接口格式

int contactless_card_close(int nHandle)

(3) 参数说明

参数	类型	说明
nHandle	int	设备句柄，open 的返回值

(4) 返回值

返回值： >= 0，成功;
<0，错误码。

(5) 错误码定义

参见 14 错误码 and 2.5 硬件错误码。

(6) 使用说明

Open 和 close 是成对的操作。当你不想再使用该设备的时候，请调用 close 来释放该设备。

query_info

(1) 功能描述

查询非接卡阅读器上卡的数目和类型。

(2) 接口格式

int contactless_card_query_info(int nHandle, unsigned int* pHasMoreCards, unsigned int* pCardType)

(3) 参数说明

参数	类型	说明
nHandle	int	设备句柄，open 的返回值
pHasMoreCards	int*	卡数目
pCardType	int*	卡类型

(4) 返回值

返回值：　　>= 0，成功;
　　<0，错误码。

(5) 错误码定义

参见 错误码 and 硬件错误码。

(6) 使用说明

pHasMoreCards:　0 : only one PICC in the field
　　0x0A : more cards in the field(type A)
　　0x0B : more cards in the field(type B)
　　0xAB : more cards in the field(type A and type B)

pCardType:

CONTACTLESS_CARD_TYPE_A_CPU	0x0000
CONTACTLESS_CARD_TYPE_B_CPU	0x0100
CONTACTLESS_CARD_TYPE_A_CLASSIC_MINI	0x0001
CONTACTLESS_CARD_TYPE_A_CLASSIC_1K	0x0002
CONTACTLESS_CARD_TYPE_A_CLASSIC_4K	0x0003
CONTACTLESS_CARD_TYPE_A_UL_64	0x0004
CONTACTLESS_CARD_TYPE_A_UL_192	0x0005
CONTACTLESS_CARD_TYPE_A_MP_2K_SL1	0x0006
CONTACTLESS_CARD_TYPE_A_MP_4K_SL1	0x0007
CONTACTLESS_CARD_TYPE_A_MP_2K_SL2	0x0008
CONTACTLESS_CARD_TYPE_A_MP_4K_SL2	0x0009

CONTACTLESS_CARD_UNKNOWN	0x00FF

attach_target

(1) 功能描述

在发送 APDU 命令前需要先为卡片上电，并获得返回值（ATR）。

(2) 接口格式

int contactless_card_attach_target(int nHandle, unsigned char* pATRBuffer, unsigned int nATRBufferLength)

(3) 参数说明

参数	类型	说明
nHandle	int	设备句柄，open 的返回值
pATRBuffer	unsigned char*	[out]ATR，空则表示无数据
nATRBufferLength	int	ATR 缓冲区的长度

(4) 返回值

返回值：　　> 0，成功, ATR 的长度;
<=0，错误码。

(5) 错误码定义

参见 错误码 and 硬件错误码。

(6) 使用说明

如果卡是 CPU 卡，则在 open 以后调用 attach_target, detach_target 和 transmit。

detach_target

(1) 功能描述

下电。如果想再次向卡发送 APDU 命令，需要再次上电。

(2) 接口格式

int contactless_card_detach_target(int nHandle)

(3) 参数说明

参数	类型	说明
nHandle	int	设备句柄，open 的返回值

(4) 返回值

返回值： >= 0，成功;
<0，错误码。

(5) 错误码定义

参见 错误码 and 硬件错误码。

(6) 使用说明

只有当你移除卡的时候才能获取返回值。

transmit

(1) 功能描述

向非接卡发送 APDU 命令，并获取答复。

(2) 接口格式

int contactless_card_transmit(int nHandle, unsigned char* pAPDU, unsigned int nAPDULength, unsigned char* pResponse, unsigned int *pResponseLength)

(3) 参数说明

parameter	type	instruction
nHandle	int	设备句柄，open 的返回值
pAPDU	unsigned char*	APDU 命令
nAPDULength	unsigned int	命令长度
pResponse	unsigned char*	APDU 的答复
pResponseLength	unsigned int	[in], 答复缓冲区的长度

		[out], 答复的长度

(4) 返回值

返回值： >= 0，成功;
<0，错误码。

(5) 错误码定义

参见 错误码 and 硬件错误码。

(6) 使用说明

非接卡的 APDU 命令和 IC 卡的 APDU 命令相同。

verify

(1) 功能描述

验证非接卡的密钥。你应该先知道密钥和密钥的类型（key A 或 key B）。

(2) 接口格式

int contactless_card_mc_verify_pin(int nHandle, unsigned int nSectorIndex, unsigned int nPinType, unsigned char* strPin, unsigned int nPinLength)

(3) 参数说明

参数	类型	说明
nHandle	int	设备句柄，open 的返回值
nSectorIndex	unsigned int	扇区索引
nPinType	unsigned int	密钥类型, 0—key A 1—key B
strPin	unsigned char*	扇区的密钥
nPinLength	unsigned int	密钥的长度

(4) 返回值

返回值： >= 0，成功;
<0，错误码。

(5) 错误码定义

参见 错误码 and 硬件错误码。

(6) 使用说明

如果卡是存储卡如 Mifare 卡，则通过 verify，read 和 write 来从卡上获取信息。

read

(1) 功能描述

从非接卡上读取信息。需要先验证密钥。

(2) 接口格式

int contactless_card_mc_read(int nHandle, unsigned int nSectorIndex, unsigned int nBlockIndex, unsigned char* pDataBuffer, unsigned int nDataBufferLength)

(3) 参数说明

参数	类型	说明
nHandle	int	设备句柄，open 的返回值
nSectorIndex	unsigned int	扇区索引
nBlockIndex	unsigned int	扇区内的块的索引
pDataBuffer	unsigned char*	数据缓冲区
nDataBufferLength	unsigned int	缓冲区的长度

(4) 返回值

返回值： >= 0，成功;
<0，错误码。

(5) 错误码定义

参见 错误码 and 硬件错误码。

(6) 使用说明

你需要从卡商那里获取卡的模式（有多少扇区、块和块的大小）。

write

(1) 功能描述

向非接卡写入数据。需要先验证密钥。

(2) 接口格式

int contactless_card_mc_write(int nHandle, unsigned int nSectorIndex, unsigned int nBlockIndex, unsigned char* pData, unsigned int nDataLength)

(3) 参数说明

参数	类型	说明
nHandle	int	设备句柄，open 的返回值
nSectorIndex	unsigned int	扇区索引
nBlockIndex	unsigned int	扇区内的块的索引
pData	unsigned char*	数据缓冲区
nDataLength	unsigned int	缓冲区的长度

(4) 返回值

返回值：　　>= 0，成功;
　　<0，错误码。

(5) 错误码定义

参见 错误码 and 硬件错误码。

write_value

(1) 功能描述

如果是电子钱包格式的卡，则向指定的块中写入钱值和用户数据。

(2) 接口格式

int contactless_card_mc_write_value(int nHandle, unsigned int nSectorIndex, unsigned int nBlockIndex, unsigned char* pValue, unsigned int nValueLength, unsigned char pAddrData)

(3) 参数说明

参数	类型	说明
nHandle	int	设备句柄，open 的返回值
nSectorIndex	unsigned int	扇区索引
nBlockIndex	unsigned int	扇区内的块的索引
pValue	unsigned char*	待写入的钱值缓冲区
nValueBufLength	unsigned int	缓冲区长度
pAddrData	unsigned char	用户数据

(4) 返回值

返回值： >= 0，成功;
<0，错误码。

(5) 错误码定义

参见 错误码 and 硬件错误码。

(6) 使用说明

该 api 将存储卡转成电子钱包卡。如果你想要使用下面的应用，需要先确认卡的类型为电子钱包卡。

read_value

(1) 功能描述

从电子钱包卡中读取钱值和用户数据

(2) 接口格式

int contactless_card_mc_read_value(int nHandle, unsigned int nSectorIndex, unsigned int nBlockIndex, unsigned char* pValue, unsigned int nValueBufLength, unsigned char* pAddrData)

(3) 参数说明

参数	类型	说明
nHandle	int	设备句柄，open 的返回值
nSectorIndex	unsigned int	扇区索引

nBlockIndex	unsigned int	扇区内的块的索引
pValue	unsigned char*	[out]读出的钱值缓冲区，4 个字节
nValueBufLength	unsigned int	缓冲区长度
pAddrData	unsigned char*	[out]用户数据,1 个字节

(4) 返回值

返回值： >= 0，成功;
<0，错误码。

(5) 错误码定义

参见 错误码 and 硬件错误码。

increment

(1) 功能描述

增加电子钱包内的钱值。该 api 与 restore 和 transfer 一起调用。

(2) 接口格式

int contactless_card_mc_increment(int nHandle, unsigned int nSectorIndex, unsigned int nBlockIndex, unsigned char* pValue, unsigned int nValueLength)

(3) 参数说明

参数	类型	说明
nHandle	int	设备句柄，open 的返回值
nSectorIndex	unsigned int	扇区索引
nBlockIndex	unsigned int	扇区内的块的索引
pValue	unsigned char*	待增加的钱值
nValueLength	unsigned int	缓冲区长度，应该至少为 4 个字节

(4) 返回值

返回值： >= 0，成功;
<0，错误码。

(5) 错误码定义

参见 错误码 and 硬件错误码。

decrement

(1) 功能描述

减少电子钱包内的钱值。该 api 与 restore 和 transfer 一起调用。

(2) 接口格式

int contactless_card_mc_decrement(int nHandle, unsigned int nSectorIndex, unsigned int nBlockIndex, unsigned char* pValue, unsigned int nValueLength)

(3) 参数说明

参数	类型	说明
nHandle	int	设备句柄，open 的返回值
nSectorIndex	unsigned int	扇区索引
nBlockIndex	unsigned int	扇区内的块的索引
pValue	unsigned char*	待减少的钱值
nValueLength	unsigned int	缓冲区长度，应该至少为 4 个字节

(4) 返回值

返回值： >= 0，成功;
<0，错误码。

(5) 错误码定义

参见 错误码 and 硬件错误码。

restore

(1) 功能描述

将钱值转存到临时寄存器中。这是 increment 和 decrement 操作的必要步骤。

(2) 接口格式

int contactless_card_mc_restore(int nHandle,unsigned int nSectorIndex, unsigned int nBlockIndex)

(3) 参数说明

参数	类型	说明
nHandle	int	设备句柄，open 的返回值
nSectorIndex	unsigned int	扇区索引
nBlockIndex	unsigned int	扇区内的块的索引

(4) 返回值

返回值：　　>= 0，成功;
　　<0，错误码。

(5) 错误码定义

参见 错误码 and 硬件错误码。

(6) 使用说明

该 api 应在 increment 或 decrement 之前调用。

transfer

(1) 功能描述

将临时寄存器内的值写回到电子钱包内。该 api 真正的修改了钱值。

(2) 接口格式

int contactless_card_mc_restore(int nHandle,unsigned int nSectorIndex, unsigned int nBlockIndex)

(3) 参数说明

参数	类型	说明
nHandle	int	设备句柄，open 的返回值
nSectorIndex	unsigned int	扇区索引

nBlockIndex	unsigned int	扇区内的块的索引

(4) 返回值

返回值： >= 0，成功;
<0，错误码。

(5) 错误码定义

参见 错误码 and 硬件错误码。

(6) 使用说明

该 api 应该在 increment 或 decrement 后调用。

2.3.3 硬件错误码

2.3.4 调用顺序

如果卡是 CPU 卡，则调用顺序为：
open→attach_target→transmit→detach_target→ close
如果卡是存储卡如 Mifare 卡，则调用顺序为：
open→ verify→read/write→ close
如果卡是电子钱包卡，则调用顺序为：
open→ verify→read_value/write_value/increment/decrement→ close
increment 和 decrement 的调用顺序为：
restore→increment/decrement→transfer。

2.4密码键盘

2.4.1 驱动文件

驱动文件名为“libUnionpayCloudPos.so”。
对应的 C 接口文件名为“pinpad_interface.h”

2.4.2 API 接口

open

(1) 功能描述

打开密码键盘设备。

(2) 接口格式

int pinpad_open()

(3) 参数说明

无参数。

(4) 返回值

返回值： >= 0，成功;
<0，错误码。

(5) 错误码定义

参见 错误码 and 硬件错误码

(6) 使用说明

该操作应该在其他操作之前被调用。

close

(1) 功能描述

关闭密码键盘设备。

(2) 接口格式

int pinpad_close()

(3) 参数说明

无参数。

(4) 返回值

返回值： >= 0，成功;
<0，错误码。

(5) 错误码定义

参见 错误码 and 硬件错误码

(6) 使用说明

Open 和 close 是成对的操作。当你不想再使用该设备的时候，请调用 close 来释放该设备。

show_text

(1) 功能描述

在 LCD 指定的行中显示信息。

(2) 接口格式

int pinpad_show_text(int nLineIndex, char* strText, int nLength, int nFlagSound)

(3) 参数说明

参数	类型	说明
nLineIndex	int	Line no to display, 0 or 1
strText	char*	Text to show, String.getBytes()
nLength	int	Text length
nFlagSound	int	Not used

(4) 返回值

返回值： >= 0，成功;
<0，错误码 , maybe your display string is too long。

(5) 错误码定义

参见 错误码 and 硬件错误码

(6) 使用说明

如果终端是 WIZARHAND_Q1，则待显示的文本为 String.getBytes()。如果密码键盘是外接设备，则待显示的中文字符参见 UserGuide-PinpadShowCharset.txt。

update_user_key

(1) 功能描述

更新会话密钥。你需要通过制定的主密钥和校验码自己校验加密的会话密钥。

(2) 接口格式

int pinpad_update_user_key(int nMasterKeyID, int nUserKeyID, unsigned char* pCipherNewUserKey, int nCipherNewUserKeyLength)

(3) 参数说明

参数	类型	说明
nMasterKeyID	int	主密钥 ID
nUserKeyID	int	用户密钥 ID 0--PIN key;1--MAC key;2—Data key
pCipherNewUserKey	unsigned char*	加密的新密钥
nCipherNewUserKeyLength	int	新密钥的长度

(4) 返回值

返回值： >= 0，成功;
<0，错误码。

(5) 错误码定义

参见 错误码 and 硬件错误码

(6) 使用说明

如果密码键盘是外接设备或者是 WIZARHAND_Q1，则通过此接口更新会话密

钥。

update_user_key_with_check

(1) 功能描述

通过校验值更新会话密钥。用户不需要自己校验用户密钥。

(2) 接口格式

int pinpad_update_user_key_with_check(int nMasterKeyID, int nUserKeyID, unsigned char *pCipherNewUserKey, int nCipherNewUserKeyLength, int nKeyUsge, unsigned char *pCheckValue, int nCheckValueLen)

(3) 参数说明

参数	类型	说明
nMasterKeyID	int	主密钥 ID
nUserKeyID	int	用户密钥 ID
pCipherNewUserKey	unsigned char*	加密的新密钥
nCipherNewUserKeyLength	int	新密钥的长度
nKeyUsge	int	密钥类型. 0--PIN key; 1--MAC key; 2—Data key
pCheckValue	unsigned char*	用户密钥的校验值
nCheckValueLen	int	校验值的长度，一般为 4 个字节。

(4) 返回值

返回值： >= 0，成功;
<0，错误码。

(5) 错误码定义

参见 错误码 and 硬件错误码

(6) 使用说明

如果终端设备是 WIZARHAND_Q1，则通过此 api 更新用户密钥。

set_pin_length

(1) 功能描述

设置密码的最大最小长度。在 calculate_pin_block 中，允许输入的密码位数不超过你设置的最大长度。

(2) 接口格式

int pinpad_set_pin_length(int nLength, int nFlag)

(3) 参数说明

参数	类型	说明
nLength	int	密码长度
nFlag	int	0—最小长度 1—最大长度

(4) 返回值

返回值：　　>= 0，成功;
　　<0，错误码。

(5) 错误码定义

参见 错误码 and 硬件错误码

(6) 使用说明

如果用户想限制输入密码的位数，则此 API 应在 calculate_pin_block 之前调用。最小长度>=4，最大长度<=12。

get_serial_number

(1) 功能描述

获得键盘的序列号。

(2) 接口格式

int pinpad_get_serial_number(unsigned char* pData,unsigned int nBufferLength)

(3) 参数说明

参数	类型	说明
pData	unsigned char*	Serial number buffer
nBufferLength	unsigned int	Length of buffer

(4) 返回值

返回值： >= 0，成功, length of serial number;
<0，错误码。

(5) 错误码定义

参见 错误码 and 硬件错误码

select_key

(1) 功能描述

在加密操作前选择一组主密钥和回话密钥。

(2) 接口格式

int pinpad_select_key(int nKeyType, int nMasterKeyID, int nUserKeyID, int nAlgorith)

(3) 参数说明

参数	类型	说明
nKeyType	int	1 : TDUKPT, 2 : MASTER-SESSION PAIR
nMasterKeyID	int	nKeyType 为 MASTER-SESSION PAIR 时，主密钥 ID， 0-9； nKeyType 为 TDUKPT,指定使用的密钥索引 0、1、2。
nUserKeyID	int	用户密钥 ID, nKeyType 为 MASTER-SESSION PAIR 时生效
nAlgorith	int	0:非国密；2:国密

(4) 返回值

返回值： >= 0，成功;
<0，错误码。

(5) 错误码定义

参见 错误码 and 硬件错误码

(6) 使用说明

当 nKeyType 为 MASTER-SESSION PAIR 时，若终端设备为 WIZARHAND_Q1，则用户密钥 ID 取值范围为 0-2，0--PIN key；1--MAC key；2—Data key；若密码键盘是外接设备，则用户密钥 ID 的取值范围为 0-1，0--PIN key；1--MAC key 或 Data key。

calculate_pin_block

(1) 功能描述

计算输入密码的 PIN block。需要先 select_key。用户密钥 ID 需要为 PIN key 对应的 ID，一般为 0。

(2) 接口格式

int pinpad_calculate_pin_block(unsigned char* pASCIICardNumber, int nCardNumberLength, unsigned char* pPinBlockBuffer, int nPinBlockBufferLength, int nTimeout_MS, int nFlagSound)

(3) 参数说明

参数	类型	说明
pASCIICardNumber	unsigned char*	卡号
nCardNumberLength	int	卡号长度
pPinBlockBuffer	unsigned char*	[out]存放 PIN block 的缓冲区
nPinBlockBufferLength	int	缓冲区长度
nTimeout_MS	int	等待用户输入时间，若小于 0，则一直等待
nFlagSound	int	未使用

(4) 返回值

返回值： >= 0，成功, length of pin block;

<0，错误码。

(5) 错误码定义

参见 错误码 and 硬件错误码

(6) 使用说明

卡号的长度一般为 13-19。

等待按键的最大时长为 60 秒，两次按键之间的最大等待时间为 10 秒，否则返回错误码。

如果是算法是国密算法，计算的 pinblock 的结果为 16 个字节；算法是 DES 或者 3DES，结果为 8 个字节。

calculate_mac

(1) 功能描述

计算 MAC。用户密钥 ID 为 MAC 可以对应的 ID，一般为 1。

(2) 接口格式

int pinpad_calculate_mac(unsigned char* pData, int nDataLength, int nMACFlag, unsigned char* pMACOutBuffer, int nMACOutBufferLength)

(3) 参数说明

参数	类型	说明
pData	unsigned char*	数据
nDataLength	int	数据长度
nMACFlag	int	0：X99， 1：ECB，2：SE919，3：银联 ECB
pMACOutBuffer	unsigned char*	[out]MAC 值缓冲区
nMACOutBufferLength	int	缓冲区长度

(4) 返回值

返回值： >= 0，成功, length of pin block;
<0，错误码。

(5) 错误码定义

参见 错误码 and 硬件错误码

encrypt_string

(1) 功能描述

加密字符串。用户密钥 ID 为 DATA key 对应的 ID，一般为 2。如果是 WizarPOS1，一般用户密钥为 select_key 是指定的 ID。

(2) 接口格式

int pinpad_encrypt_string(unsigned char* pPlainText, int nTextLength, unsigned char* pCipherTextBuffer, int nCipherTextBufferLength)

(3) 参数说明

参数	类型	说明
pPlainText	unsigned char*	字符串明码
nTextLength	int	字符串长度
pCipherTextBuffer	unsigned char*	[out]密码字符串的缓冲区
nCipherTextBufferLength	int	缓冲区长度

(4) 返回值

返回值： >= 0，成功, length of pin block;
<0，错误码。

(5) 错误码定义

参见 错误码 and 硬件错误码

(6) 使用说明

如果是国密算法，待加密的数据必须是 16 个字节，不足 16 字节的，用户自己补足。加密结果也是 16 个字节。加密是默认的模式是 ECB，NoPadding。

update_cipher_master_key

(1) 功能描述

更新密文主密钥，主密钥通过传输密钥加密，传输密钥索引和主密钥索引相同。

(2) 接口格式

int pinpad_update_cipher_master_key (int nMasterKeyID,
unsigned char* pCipherMasterKey, int nCipherMasterKeyLen,
unsigned char *pCheckValue, int nCheckValueLen)

(3) 参数说明

参数	类型	说明
nMasterKeyID	int	主密钥索引
pCipherMasterKey	unsigned char*	密文主密钥
nCipherMasterKeyLen	int	主密钥长度
pCheckValue	unsigned char*	校验值
nCheckValueLen	int	校验值长度

(4) 返回值

返回值： >= 0，成功;
<0，错误码。

(5) 错误码定义

参见 错误码 and 硬件错误码

update_user_key_with_check_E

(1) 功能描述

更新工作密钥，采用指定的校验算法。

(2) 接口格式

int pinpad_update_user_key_with_check_E (int nMasterKeyID, int nUserKeyID,
unsigned char *pCipherNewUserKey, int nCipherNewUserKeyLength,
int nKeyUsge, unsigned char *pCheckValue, int nCheckValueLen,

int algoCheckValue)

(3) 参数说明

参数	类型	说明
nMasterKeyID	int	主密钥 ID
nUserKeyID	int	用户密钥 ID
pCipherNewUserKey	unsigned char*	加密的新密钥
nCipherNewUserKeyLength	int	新密钥的长度
nKeyUsge	int	密钥类型. 0--PIN key; 1--MAC key; 2—Data key
pCheckValue	unsigned char*	用户密钥的校验值
nCheckValueLen	int	校验值的长度，一般为 4 个字节。
algoCheckValue	int	ALGO_CHECK_VALUE 类型 0--ALGO_CHECK_VALUE_DEFAULT 1--ALGO_CHECK_VALUE_SE919

(4) 返回值

返回值： >= 0，成功;
<0，错误码。

(5) 错误码定义

参见 错误码 and 硬件错误码

update_cipher_master_key_E

(1) 功能描述

更新密文主密钥，密文主密钥采用传输密钥加密，传输密钥索引和主密钥索引相同，校验算法通过参数设置。

(2) 接口格式

int pinpad_update_cipher_master_key_E (int nMasterKeyID,
unsigned char *pCipherMasterKey, int nCipherMasterKeyLen,
unsigned char *pCheckValue, int nCheckValueLen,
int algoCheckValue)

(3) 参数说明

参数	类型	说明
nMasterKeyID	int	主密钥索引
pCipherMasterKey	unsigned char*	密文主密钥
nCipherMasterKeyLen	int	主密钥长度
pCheckValue	unsigned char*	校验值
nCheckValueLen	int	校验值长度
algoCheckValue	int	ALGO_CHECK_VALUE 类型 0--ALGO_CHECK_VALUE_DEFAULT 1--ALGO_CHECK_VALUE_SE919

(4) 返回值

返回值： >= 0，成功;
<0，错误码。

(5) 错误码定义

参见 错误码 and 硬件错误码

encrypt_string_with_mode

(1) 功能描述

加密数据时指定模式，初始向量及初始向量长度。

(2) 接口格式

int pinpad_encrypt_string_with_mode(unsigned char* pPlainText, int nTextLength, unsigned char* pCipherTextBuffer, int nCipherTextBufferLength,
unsigned int nMode, unsigned char* pIV, unsigned int nIVLen);

(3) 参数说明

参数	类型	说明
pPlainText	unsigned char*	字符串明码
nTextLength	int	字符串明码长度
pCipherTextBuffer	unsigned char*	密码字符串的缓冲区，如果是 dukpt 加密，那么数据末尾还包含 KSN 及 Counter。
nCipherTextBufferLength	int	缓冲区长度

nMode	int	加密模式： 0 --PINPAD_ENCRYPT_STRING_MODE_EBC 1--PINPAD_ENCRYPT_STRING_MODE_CBC 2--PINPAD_ENCRYPT_STRING_MODE_CFB 3--PINPAD_ENCRYPT_STRING_MODE_OFB Padding 使用的是 NoPadding。
pIV	unsigned char*	初始向量，模式为 CBC，CFB，OFB 时设置有效
nIVLen	int	初始向量长度

(4) 返回值

返回值： >= 0，成功; 返回值长度为加密后的数据长度。
<0，错误码。

(5) 错误码定义

参见 错误码 and 硬件错误码

2.4.3 硬件错误码

错误名称	取值	说明
SUCCESS	0x00	success
EACCES	0x11	access denied
CMD_MISMATCH	0x12	wrong command id
PACKAGE_LENGTH	0x13	wrong package length
USER_CANCEL	0x01	user cancel
FIELD_LENGTH	0x15	wrong length of field
NOT_SUPPORT	0x20	This function is not supported
PIN_LENGTH	0x21	out of range of pin length
AUTH	0x22	failed in authentication
KEY_LENGTH	0x23	wrong length of key
CHECK_VALUE	0x24	wrong check value of session key
WRITE_FLASH	0x25	failed in writing flash
READ_FLASH	0x26	failed in reading flash
NO_KEY	0x27	no key in this field
OUT_RANGE	0x28	input is out legal range
KEY_ERROR_INTEGRITY	0x29	failed in checking integrity
AES_ENCRYPT	0x2A	failed in encrypting using aes key text
AES_DECRYPT	0x2B	failed in decrypting using aes key
BREAK_SENSITIVE_RULES	0x2C	break rules about data sensitivity

MAC	0x2D	failed in the process of calculating mac
DATA_ALIGN	0x2E	data length is not aligned

2.4.4 调用顺序

open→show_text/…/get_serial_number →close
或
open→select_key→calculate_pin_block/.calculate_mac/encrypt_string→close

2.5打印机

2.5.1 驱动文件

驱动文件名为“libUnionpayCloudPos.so”。
对应的 C 接口文件名为“printer_interface.h”

2.5.2 API 接口

open

(1) 功能描述

打开打印机设备。

(2) 接口格式

int printer_open()

(3) 参数说明

无参数。

(4) 返回值

返回值： >= 0，成功;
<0，错误码。

(5) 错误码定义

参见 错误码 and 硬件错误码。

(6) 使用说明

此操作应该在其他操作之前调用。Open 和 begin 操作一般一起调用。

begin

(1) 功能描述

准备打印。

(2) 接口格式

int printer_begin()

(3) 参数说明

无参数。

(4) 返回值

返回值： >= 0，成功;
<0，错误码。

(5) 错误码定义

参见 错误码 and 硬件错误码。

(6) 使用说明

在这个 API 后可以打印数据。
通过 endAPI 终止该操作。

end

(1) 功能描述

停止打印。

(2) 接口格式

int printer_end()

(3) 参数说明

无参数。

(4) 返回值

返回值： >= 0，成功;
<0，错误码。

(5) 错误码定义

参见 错误码 and 硬件错误码。

(6) 使用说明

Begin 和 end 是成对的操作。

close

(1) 功能描述

关闭设备。

(2) 接口格式

int printer_close()

(3) 参数说明

无参数。

(4) 返回值

返回值： >= 0，成功;
<0，错误码。

(5) 错误码定义

参见 错误码 and 硬件错误码。

(6) 使用说明

Open 和 close 是成对的操作。当你不想再使用该设备的时候，请调用 close 来释放该设备。

query_status

(1) 功能描述

查询打印机状态。

(2) 接口格式

int printer_query_status()

(3) 参数说明

无参数。

(4) 返回值

返回值： == 1, has paper。
== 0，成功;
< 0，错误码。

(5) 错误码定义

参见 错误码 and 硬件错误码。

(6) 使用说明

此 API 应该只 open 和 close 之间调用，但不在 begin 和 end 之间。

This api should be used inner open and close, but not inner begin and end。

write

(1) 功能描述

打印数据。数据可以是字符串或图片。

(2) 接口格式

int printer_write(unsigned char* pData, int nDataLength)

(3) 参数说明

参数	类型	说明
pData	unsigned char*	Data or control command
nDataLength	int	Length of data

(4) 返回值

返回值：　　== 1, has paper。
== 0，成功;
< 0，错误码。

(5) 错误码定义

参见 错误码 and 硬件错误码。

2.5.3 硬件错误码

2.5.4 调用顺序

open→query_status→begin→write→end→ close

2.6LED

2.6.1 驱动文件

驱动文件名为“libUnionpayCloudPos.so”。
对应的 C 接口文件名为“led_service_interface.h”

2.6.2 API 接口

open

(1) 功能描述

打开 LED 设备，包含所有的 led 灯的服务。

(2) 接口格式

int led_open()

(3) 参数说明

无参数。

(4) 返回值

返回值：　　>= 0，成功;
　　<0，错误码。

(5) 错误码定义

参见 错误码 and 硬件错误码。

(6) 使用说明

此 API 应在其它操作之前调用。

close

(1) 功能描述

关闭 LED 设备，包含所有的 LED 灯的服务

(2) 接口格式

int led_close()

(3) 参数说明

无参数。

(4) 返回值

返回值： >= 0，成功;
<0，错误码。

(5) 错误码定义

参见 错误码 and 硬件错误码。

(6) 使用说明

Open 和 close 是成对的操作。当你不想再使用该设备的时候，请调用 close 来释放该设备。

turn_on

(1) 功能描述

点亮指定的 LED 灯。

(2) 接口格式

int led_on(unsigned int nLedIndex)

(3) 参数说明

参数	类型	说明
nLedIndex	unsigned int	Index of led, >= 0 && < MAX_LED_COUNT

(4) 返回值

返回值： >= 0，成功;
<0，错误码。

(5) 错误码定义

参见 错误码 and 硬件错误码。

turn_off

(1) 功能描述

熄灭指定的 LED 灯。

(2) 接口格式

int led_off(unsigned int nLedIndex)

(3) 参数说明

参数	类型	说明
nLedIndex	unsigned int	Index of led, >= 0 && < MAX_LED_COUNT

(4) 返回值

返回值： >= 0，成功;
<0，错误码。

(5) 错误码定义

参见 错误码 and 硬件错误码。

get_status

(1) 功能描述

获得指定 LED 灯的状态。

(2) 接口格式

int led_get_status(unsigned int nLedIndex)

(3) 参数说明

参数	类型	说明
nLedIndex	unsigned int	Index of led, >= 0 && < MAX_LED_COUNT

(4) 返回值

返回值： > 0, LED on;
= 0, LED off;
< 0，错误码。

(5) 错误码定义

参见 错误码 and 硬件错误码。

2.6.3 硬件错误码

2.6.4 调用顺序

open→turn_on/turn_off/get_status → close

2.7串口

2.7.1 驱动文件

驱动文件名为“libUnionpayCloudPos.so”。
对应的 C 接口文件名为“ext_serial_port_interface.h”

2.7.2 API 接口

open

(1) 功能描述

通过制定的设备名称打开串口设备。

(2) 接口格式

int esp_open(char* pDeviceName)

(3) 参数说明

参数	类型	说明
pDeviceName	char*	设备名称

(4) 返回值

返回值： >= 0, handle of this device;
<0，错误码。

(5) 错误码定义

参见 错误码 and 硬件错误码。

(6) 使用说明

此操作应该在其他操作之前调用。
CLOUDPOS 默认的 pDeviceName 为“/dev/s3c2410_serial2”

close

(1) 功能描述

关闭之前打开的串口设备。

(2) 接口格式

int esp_close(int nHandle)

(3) 参数说明

参数	类型	说明
nHandle	int	设备句柄，open 的返回值

(4) 返回值

返回值： >= 0，成功;
<0，错误码。

(5) 错误码定义

参见 错误码 and 硬件错误码。

(6) 使用说明

Open 和 close 是成对的操作。当你不想再使用该设备的时候，请调用 close 来释放该设备。

set_baudrate

(1) 功能描述

设置串口的波特率使得读写操作能够在相同的波特率下进行。

(2) 接口格式

int esp_set_baudrate(int nHandle, unsigned int nBaudrate)

(3) 参数说明

参数	类型	说明
nHandle	int	设备句柄，open 的返回值
nBaudrate	int	波特率

(4) 返回值

返回值： > 0，成功;
<=0，错误码。

(5) 错误码定义

参见 错误码 and 硬件错误码。

(6) 使用说明

此 API 应在 read 和 write 之前调用。

read

(1) 功能描述

从串口设备中获得数据。

(2) 接口格式

int esp_read(int nHandle, unsigned char* pDataBuffer, int nExpectedDataLength, int nTimeout_MS)

(3) 参数说明

参数	类型	说明
nHandle	int	设备句柄，open 的返回值
pDataBuffer	unsigned char*	数据缓冲区
nExpectedDataLength	int	缓冲区长度
nTimeout_MS	int	超时时间. 0：读并立即返回 <0：直到读到数据才返回

(4) 返回值

返回值： >= 0, 数据长度;
<0，错误码。

(5) 错误码定义

参见 错误码 and 硬件错误码。

write

(1) 功能描述

向串口设备传送数据。

(2) 接口格式

int esp_write(int nHandle, unsigned char* pDataBuffer, int nDataLength)

(3) 参数说明

参数	类型	说明
nHandle	int	设备句柄，open 的返回值
pDataBuffer	unsigned char*	数据缓冲区
nDataLength	int	缓冲区长度

(4) 返回值

返回值： >= 0, 写入的数据长度;
<0，错误码。

(5) 错误码定义

参见 错误码 and 硬件错误码。

flush_io

(1) 功能描述

清空串口的 IO 缓冲区。

(2) 接口格式

int esp_flush_io(int nHandle)

(3) 参数说明

参数	类型	说明
nHandle	int	设备句柄，open 的返回值

(4) 返回值

返回值：　　>= 0，成功;
　　<0，错误码。

(5) 错误码定义

参见 错误码 and 硬件错误码。

setEnable

(1) 功能描述

启用和禁用该硬件。

(2) 接口格式

int esp_setEnable(unsigned int nEnable)

(3) 参数说明

参数	类型	说明
nEnable	unsigned int	1 : enable, 0 : disable。

(4) 返回值

返回值：　　>= 0，成功;
　　<0，错误码。

(5) 错误码定义

参见 错误码 and 硬件错误码。

2.7.3 硬件错误码

2.7.4 调用顺序

open→set_baudrate→read/write→fulsh_io→ close

2.8客显

2.8.1 驱动文件

驱动文件名为“libUnionpayCloudPos.so”。
对应的 C 接口文件名为“customer_display_interface.h”

2.8.2 API 接口

open

(1) 功能描述

打开客显设备。

(2) 接口格式

void* customer_display_open(int* pError)

(3) 参数说明

参数	类型	说明
pErrorCode	int*	[out]错误码，如果返回值为 0

(4) 返回值

返回值：　　其它，成功，值为设备句柄;
0，错误码。

(5) 错误码定义

参见 错误码 and 硬件错误码。

(6) 使用说明

此 API 应在其它操作之前调用。

close

(1) 功能描述

关闭打开的客显设备。

(2) 接口格式

int customer_display_close(int nHandle)

(3) 参数说明

参数	类型	说明
nHandle	int	设备句柄，open 的返回值

(4) 返回值

返回值： >= 0，成功;
<0，错误码。

(5) 错误码定义

参见 错误码 and 硬件错误码。

(6) 使用说明

Open 和 close 是成对的操作。当你不想再使用该设备的时候，请调用 close 来释放该设备。

ctrl_devs

(1) 功能描述

使用此方法来设置客显的背景、默认显示、打开或关闭 LED 以及蜂鸣器灯。

(2) 接口格式

int customer_display_ctrl_devs(int nHandle, int nCmd, int nValue)

(3) 参数说明

参数	类型	说明
nHandle	int	设备句柄，open 的返回值
nCmd	int	控制命令
nValue	int	控制命令的取值

(4) 返回值

返回值：　　>= 0，成功;
　　<0，错误码。

(5) 错误码定义

参见 错误码 and 硬件错误码。

(6) 使用说明

nCmd	nValue
0x04:set background。	RED :0x001F; BLACK:0X0000; YELLOW:0X07FF; BLUE:0XF800; GRAY0:0XCE9A。
0x05:display default screen。	0
0x06:led。	1:打开 led; 0:关闭 led。
0x07:buzzer。	0

write_picture

(1) 功能描述

画图（ARGB8888 模式。）

(2) 接口格式

int customer_display_write_picture_data(int nHandle, unsigned int nXcoordinate, unsigned int nYcoordinate,unsigned int nWidth, unsigned int nHeight ,unsigned char* pData, unsigned int nDataLength)

(3) 参数说明

参数	类型	说明
nHandle	int	设备句柄，open 的返回值
nXcoordinate	unsigned int	X 坐标
nYcoordinate	unsigned int	Y 坐标
nWidth	unsigned int	宽度
nHeight	unsigned int	高度
pData	unsigned char*	全部点数据
nDataLength	unsigned int	数据长度

(4) 返回值

返回值： >= 0，成功;
<0，错误码。

(5) 错误码定义

参见 错误码 and 硬件错误码。

(6) 使用说明

需要首先把图片转成点数据。

2.8.3 硬件错误码

2.8.4 调用顺序

open→ctrl_devs/write_picture→ close

2.9身份证阅读器

2.9.1 驱动文件

驱动文件名为“libUnionpayCloudPos.so”。
对应的 C 接口文件名为“customer_display_interface.h”

2.9.2 API 接口

open

(1) 功能描述

打开身份证明阅读器设备。

(2) 接口格式

int identity_card_open()

(3) 参数说明

无参数。

(4) 返回值

返回值： >= 0, handle of this device;
<0，错误码。

(5) 错误码定义

参见 错误码 and 硬件错误码。

(6) 使用说明

此操作应该在其他操作之前调用。

close

(1) 功能描述

关闭之前打开的身份证阅读器设备。

(2) 接口格式

int identity_card_close(int nHandle);

(3) 参数说明

参数	类型	说明
nHandle	int	设备句柄，open 的返回值

(4) 返回值

返回值： >= 0，成功;
<0，错误码。

(5) 错误码定义

参见 错误码 and 硬件错误码。

(6) 使用说明

Open 和 close 是成对的操作。当你不想再使用该设备的时候，请调用 close 来释放该设备。

search_target

(1) 功能描述

开始寻找身份证。此方法直到寻到身份证才返回。

(2) 接口格式

int identity_card_search_target(int nHandle)

(3) 参数说明

参数	类型	说明
nHandle	int	设备句柄，open 的返回值

(4) 返回值

返回值： 0，失败;
其它，错误码。

(5) 错误码定义

参见 错误码 and 硬件错误码。

get_information

(1) 功能描述

获得身份证的信息。

(2) 接口格式

int identity_card_get_fixed_information(int nHandle, struct tagIDCardData *pIDCardData)

(3) 参数说明

参数	类型	说明
nHandle	int	设备句柄，open 的返回值
pIDCardData	struct tagIDCardData *	身份证数据

(4) 返回值

返回值： >= 0，成功，值为图片数组的长度;
<0，错误码。

(5) 错误码定义

参见 错误码 and 硬件错误码。

(6) 使用说明

```
typedef struct tagIDCardData
{
    unsigned char strName[30];
    unsigned char strSex[2];
    unsigned char strNation[4];
    unsigned char strBorn[16];
    unsigned char strAddress[70];
    unsigned char strIDCardNo[36];
    unsigned char strGrantDept[30];
    unsigned char strUserLifeBegin[16];
    unsigned char strUserLifeEnd[16];
```

```
    unsigned char strReserved[36];
    unsigned char strPicture[1024];
}IDCARD_PROPERTY;
```

2.9.3 硬件错误码

2.9.4 调用顺序

open→search_target→get_information→ close

2.10钱箱

2.10.1 驱动文件

驱动文件名为“libUnionpayCloudPos.so”。
对应的 C 接口文件名为“moneybox_interface.h”

2.10.2 API 接口

open

(1) 功能描述

打开钱箱设备。

(2) 接口格式

int moneybox_open()

(3) 参数说明

无参数。

(4) 返回值

返回值：　　>= 0，成功;
　　<0，错误码。

(5) 错误码定义

参见 错误码 and 硬件错误码。

(6) 使用说明

此操作应该在其他操作之前调用。

close

(1) 功能描述

关闭钱箱设备。

(2) 接口格式

int moneybox_close()

(3) 参数说明

无参数。

(4) 返回值

返回值： >= 0，成功;
<0，错误码。

(5) 错误码定义

参见 错误码 and 硬件错误码。

(6) 使用说明

Open 和 close 是成对的操作。当你不想再使用该设备的时候，请调用 close 来释放该设备。

kick_out

(1) 功能描述

弹出钱箱。

(2) 接口格式

int moneybox_ctrl()

(3) 参数说明

无参数。

(4) 返回值

返回值：　　>= 0，成功;
　　<0，错误码。

(5) 错误码定义

参见 错误码 and 硬件错误码。

setEnable

(1) 功能描述

启用和禁用该硬件。

(2) 接口格式

int moneybox_setEnable(unsigned int nEnable)

(3) 参数说明

参数	类型	说明
nEnable	unsigned int	1 : enable, 0 : disable。

(4) 返回值

返回值： >= 0，成功;
<0，错误码。

(5) 错误码定义

参见 错误码 and 硬件错误码。

2.10.3 硬件错误码

2.10.4 调用顺序

open→kick_out→ close

2.11 键盘

2.11.1 键盘特殊键值

CLOUDPOS 系列机型，键盘上存在一些特殊按键，这些特殊按键的键值是自定义的。下面分别介绍不同机型下，特殊按键的键值对应关系。

Q1 或 M0

物理按键	键值	描述
句号键	输入法是中文时，键值为 "。"；输入法是英文时，键值为 KeyEvent.KEYCODE_PERIOD	
二维码键	232	
回退键（红色 X）	KeyEvent.KEYCODE_ESCAPE	
扫描键左	229	
扫描键右	230	仅在 M0 上存在。
通话键	231	仅在 M0 上存在。
del 键(黄色三角块)	KeyEvent.KEYCODE_DEL	
回车键（绿色圆）	KeyEvent.KEYCODE_ENTER	

其他键值都是标准的。参见 android.view.KeyEvent

PAD 或 POS

键值都是标准的。参见 android.view.KeyEvent

2.11.2 扫描键和二维码键

2.11.3 用法说明

扫描快捷按键，在 Q1 上是位于机器屏幕左边的橙色按钮，在 M0 上是位于机器屏幕两边的橙色按钮。

二维码键是位于 Q1 或 M0 物理键盘最下方中间的橙色按键。

这两种按键具有快速启动扫描应用的功能。这个快速启动的过程由系统提供。但需要扫描应用在 android manifest 文件中加入一组 intent-filter，其中 category 和 action 如下所示：

<action android:name="android.intent.action.SCAN" />

<category android:name="android.intent.category.DEFAULT" />

当按下扫描键或二维码键事件发生时，系统会去寻找包含这组 intent-filter 的应用并启动该应用，如果有多个这样的应用，那么提供应用列表，供用户选择要启动的应用。如果没有任何应用配置这样的 intent-filter，按键不会启动扫描。

2.11.4 使用范例

如下示范了扫描键或二维码键的两种使用方式：

1. 如果在按下扫描键或二维码键时快速启动应用，那么可以在应用的 android manifest 文件中声明 category 和 action，如下所示：

```
<activity
        android:name="com.XX.activity.MainActivity"
        android:label="@string/app_name" >
        <intent-filter>
                <!-- 设置当有人按下 scan 按键后调用扫描应用 -->
                <action android:name="android.intent.action.SCAN" />
                <category android:name="android.intent.category.DEFAULT" />
        </intent-filter>
        </activity>
```

2. 如果在按下扫描键或二维码键时，仅做一些处理逻辑，那么这个过程和其他普通按键的方式是一样的，必须重写 onKeydown 方法，应用内捕获扫描键或二维码键，此时的应用必须是激活状态的，代码如下所示：

```
 /** Key code constant: left scan key. */
public static final int KEYCODE_SCAN_LEFT               = 229;

 /** Key code constant: right scan key. */
public static final int KEYCODE_SCAN_RIGHT              = 230;

/** Key code constant: qr key. */
public static final int KEYCODE_QR                      = 232;

public boolean onKeyDown(int keyCode, KeyEvent event) {
        // 监听物理键
        if (keyCode == KEYCODE_SCAN_LEFT|| keyCode ==KEYCODE_SCAN_RIGHT|| keyCode ==KEYCODE_QR) {
```

```
            //处理逻辑，比如扫描。。。。。。
            return true;
        }
        return super.onKeyDown(keyCode, event);
    }
```

2.12指纹

2.12.1 驱动文件

驱动文件名为“libUnionpayCloudPos.so”。
对应的 C 接口文件名为“fingerprint_interface.h”

2.12.2 API 接口

open

(1) 功能描述

打开指纹设备。

(2) 接口格式

int fp_open ()

(3) 参数说明

无参数。

(4) 返回值

返回值：　　>= 0，成功；
　　<0，错误码。

(5) 错误码定义

参见 错误码 and 硬件错误码。

(6) 使用说明

此操作应该在其他操作之前调用。

close

(1) 功能描述

关闭之前打开的指纹设备。

(2) 接口格式

int fp_close ();

(3) 参数说明

(4) 返回值

返回值： >= 0，成功;
<0，错误码。

(5) 错误码定义

参见 错误码 and 硬件错误码。

(6) 使用说明

Open 和 close 是成对的操作。当你不想再使用该设备的时候，请调用 close 来释放该设备。

get_fea

(1)功能描述

获取指纹特征值。

(2) 接口格式

int fp_get_fea(unsigned char *pFeaBuffer, int nFeaLength, int *pRealFeaLength, int n_TimeOut_S)

(3) 参数说明

参数	类型	说明
pFeaBuffer	char *	传入的存储特征值的 buffer，不可为 NULL
nFeaLength	int	pFeaBuffer 的长度
pRealFeaLength	int *	返回的特征值数据长度
nTimeOut_S	int	超时时间，单位：秒

(4) 返回值

返回值： >= 0，成功；
<0，错误码。

(5) 错误码定义

参见 错误码 and 硬件错误码。

(6) 使用说明

手指触碰指纹位置，接口会返回指纹特征值。

getLastImage

(1)功能描述

获取指纹图像。

(2) 接口格式

int fp_getLastImage(unsigned char *pImgBuffer,int nImgLength, int *pRealImaLength, int *pImgWidth, int *pImgHeight)

(3) 参数说明

参数	类型	说明
pImgBuffer	char *	图像 Buffer
nImgLength	int	pImgBuffer 长度
pRealImaLength	int *	返回实际获取的图像数据长度
pImgWidth	int *	返回图像的宽度
pImgHeight	int *	返回图像的高度

(4) 返回值

返回值： >= 0，成功；
<0，错误码。

(5) 错误码定义

参见 错误码 and 硬件错误码。

(6) 使用说明

手指触碰指纹位置，接口会返回指纹图像。

match

(1)功能描述

指纹比对。

(2) 接口格式

int fp_match(unsigned char *pFeaBuffer1, int nFea1Length, unsigned char *pFealBuffer2, int nFea2Length)

(3) 参数说明

参数	类型	说明
pFeaBuffer1	char *	特征值 1
nFea1Length	int	特征值 1 数据长度
pFeaBuffer2	char *	特征值 2

nFea2Lenghth	int	特征值 2 数据长度

(4) 返回值

返回值： >0, 比对相似度百分比；
<=0，错误码。

(5) 错误码定义

参见 错误码 and 硬件错误码。

(6) 使用说明

比较两个指纹特征值。

cancel

(1)功能描述

取消当前进行功能，例如取消正在进行获取特征值操作。

(2) 接口格式

int fp_cancel ()

(3) 参数说明

无参数。

(4) 返回值

返回值： >= 0, 成功；
<0，错误码。

(5) 错误码定义

参见 错误码 and 硬件错误码。

2.13 上网 APN 配置

在终端上支持手工配置 APN，配置方式如下：
设置-》移动数据-》接入点名称（APN），可以添加新的 APN 设置。
同时提供一套接口，应用程序可以通过这些接口添加新的 APN。

2.13.1 API 接口

addByAllArgs

(1) 功能描述

添加新的 APN。

(2) 接口格式

String addByAllArgs(String name, String apn, String mcc, String mnc, String proxy, String port, String MMSProxy, String MMSPort, String userName, String server, String password, String MMSC, String authType, String protocol, String roamingProtocol, String type, String bearer, String MVNOType, String MVNOMatchData)

(3) 参数说明

参数	类型	说明
name	String	名称,必传
apn	String	APN,必传
mcc	String	MCC,不可为空
mnc	String	MNC,不可为空
proxy	String	
port	String	
MMSProxy	String	彩信代理,可为空
MMSPort	String	彩信端口,可为空
userName	String	
server	String	
password	String	
MMSC	String	
authType	String	身份验证类型(无,PAP,CHAP,PAP/CHAP),可为空,默认为无
protocol	String	APN 协议(IPV4,IPV6,IPV4/IPV6),可为空,默认为 IPV4
roamingProtocol	String	APN 漫游协议(IPV4,IPV6,IPV4/IPV6),可为空,默认为 IPV4

type	String	APN 类型,可为空
bearer	String	承载系统(LTE,eHRPD,无),可为空,默认为无
MVNOType	String	MVNO 类型(无,SPN,IMSI,GID),可为空,默认为无
MVNOMatchData	String	

(4) 返回值

返回值： “succeed”，成功;
“详细的错误提示信息”，失败。

add

(1) 功能描述

添加新的 APN。

(2) 接口格式

String add(String name, String apn)

(3) 参数说明

参数	类型	说明
name	String	名称,必传
apn	String	APN,必传

(4) 返回值

返回值： “succeed”，成功;
“详细的错误提示信息”，失败。

addByMCCAndMNC

(1) 功能描述

添加新的 APN。

(2) 接口格式

String addByMCCAndMNC(String name, String apn, String mcc, String mnc)

(3) 参数说明

参数	类型	说明
name	String	名称,必传
apn	String	APN,必传
mcc	String	MCC,不可为空
mnc	String	MNC,不可为空

(4) 返回值

返回值： “succeed”，成功;
“详细的错误提示信息”，失败。

setSelected

(1) 功能描述

设置默认选中的 APN。

(2) 接口格式

boolean setSelected(String name)

(3) 参数说明

参数	类型	说明
name	String	名称,必传

(4) 返回值

返回值： true，成功;
false，失败。

clear

(1) 功能描述

清除所有 APN 设置。

(2) 接口格式

boolean clear()

(3) 参数说明

无参数。

(4) 返回值

返回值： true，成功;
false，失败。

2.13.2 用法说明

ＡＰＮ接口的类是 com.wizarpos.wizarviewagentassistant.APNManagerService，由系统提供， 使用的时候，导入所在的包就可以了。另外，必须配置需要的权限，见 14.2.11 节。

2.14参数下载完成处理

第三方参数服务是一个远程依托于 wizarview，本地基于终端系统应用 wizaragent 的服务，主要目的是方便外部合作伙伴更加便捷的管理和使用慧银智能终端的服务，通过它可以向其他第三方应用远程推送第三方信息。
第三方应用通过 Android 系统的广播来得知参数已经下载。

2.14.1 流程

注册参数下发广播接收器

Wizarview 将操作员所编写的参数信息，上传并下载，最后第三方应用通过注册特定接收器来获取远程 wizarview 所推送的文本信息。

代码示例：

```
//使用 android receiver  获得推送通知。
<!-- 第三方推送权限 -->
<uses-permission android:name="android.permission.CLOUDPOS_WIZARVIEW"/>


<!-- 对监听第三方推送 -->
<receiver android:name="com.xxx.xxx.receiver.XXBroadcastReceiver"
        android:permission="android.permission.CLOUDPOS_DOWNLOADRVICE" >
    <intent-filter>
        <action android:name="第三方应用的包名" />
    </intent-filter>
</receiver>

public class ParamFileReceiver extends BroadcastReceiver {

        private static final String MSG_TYPE_PARAM = "param:";
        @Override
        public void onReceive(Context context, Intent intent) {
                String notification = intent.getStringExtra("notification");
                if(notification != null&& MSG_TYPE_PARAM. Equals(
                        notification.subSequence(0, MSG_TYPE_PARAM.length()))){
                        String fileName = notification.substring(MSG_TYPE_PARAM.length());
                        //去除多余的字符：以 TYPE 长度截取参数
                }
        }
}
```

注：

注意推送信息会有一个类型字段，用于区分第三方信息推送。

下载参数信息

获取参数文件名，通过 ContentResolver 和指定的 uri 来获取已经下载成功的参数文件，用于配置。

代码示例：

//通过监听广播获得参数文件名，然后使用 ContentResolver 来获得文件流。

```
URI_PARAM_FILE = "content://com.wizarpos.wizarviewagent.paramfilesprovider/file/";
Uri uri = Uri.parse(ParamFileProvider.URI_PARAM_FILE + fileName);
ContentResolver resolver = context.getContentResolver();
Reader reader = null;
try {
    reader = new InputStreamReader(resolver.openInputStream(uri));
} catch (FileNotFoundException e) {
    e.printStackTrace();
}
```

通过这个 reader 来完成读取参数文件。

参数信息回调

通知服务器已应用该参数或无法应用该参数，当无法应用参数时，可以将错误信息利用 ContentResolver 来反馈给 wizarview 服务器。

代码示例：

```
//参数文件是否完成，用于完成或终止参数推送。
// 表示应用是否已经应用 或适配。
public static final String KEY_READED = "readed";
// 如果应用失败，可以上传错误信息。
public static final String KEY_ERRLOG = "errlog";
String URI_PARAM_FILE
  = "content://com.wizarpos.wizarviewagent.paramfilesprovider/file/";
```

```
Uri uri = Uri.parse(ParamFileProvider.URI_PARAM_FILE + fileName);
ContentResolver resolver = context.getContentResolver();
ContentValues vaules = new ContentValues();
// 代表参数信息是已经已经应用成功。
vaules.put("readed", true);

// 代表参数信息是无法应用。
vaules.put("readed", false);
vaules.put("errlog", "参数信息有错误，无法应用");

// 将参数适配的结果通知给服务
Uri resultUri = resolver.insert(uri, vaules);

if(resultUri == null){
    Log.e(APP_TAG, "param happen error , you need check log for modify this       question!");
 }
```

2.15W1 上屏蔽 home 键

一般情况下，home 键由系统负责监听，捕获后由系统进行处理，但实际情况是，一些应用由于业务逻辑的要求，需要屏蔽/捕获 home 键，自己的应用出来处理。
以下定义了两种方式：

2.15.1 指定整个 apk 屏蔽/捕获 home 键

通过申明 android.permission.CLOUDPOS_DISABLE_HOME_KEY 权限，应用就会捕获 home 键自己处理。

2.15.2 指定某个 activity 捕获 home 键

通过申明 android.permission.CLOUDPOS_DISABLE_HOME_KEY_IN_ACTIVITY 权限，表示终端有权限去屏蔽/捕获 home 键，但是应用中还需要做一些处理，在不同的终端型号中处理方式不同：

在 W1 中，需要设置 WindowManager.LayoutParams.TYPE_KEYGUARD 或者 TYPE_KEYGUARD_DIALOG。

示例代码如下：

```
public void onAttachedToWindow() {
        super.onAttachedToWindow();
        try {
                this.getWindow().setType(WindowManager.LayoutParams.TYPE_KEYGUARD);
        } catch (Throwable e) {
                e.printStackTrace();
        }

}
```

2.16 Q1 上的全屏处理

2.16.1 使用 Android 方式

在 Q1 上实现应用全屏的效果，需要做如下的处理：

1. 申明 android.permission.CLOUDPOS_REAL_FULLSCREEN 权限。
2. 通过 getWindow().getDecorView().setSystemUiVisibility（）这个方法设置沉浸式全屏模式。用户可以自由组合参数来实现是否隐藏状态栏，是否隐藏通知栏。

由于 Q1 没有与状态栏上的按钮对应的物理按键，屏蔽状态栏则意味着：底部 home 键不起效并转发给应用；recent 按键失效；下拉状态栏失效；上推导航栏失效。需要应用在自己的界面中去设置相同含义的 button，自行处理。

但在 Q1 的物理键盘上左下角的 X 键意义和 Back 键的含义相同，因此，客户应用可以捕获到这个键值，处理 Back 的逻辑。

示例代码如下：

```
//statusBar，navigationBar 都不显示
 getWindow().getDecorView().setSystemUiVisibility(
View.SYSTEM_UI_FLAG_HIDE_NAVIGATION
| View.SYSTEM_UI_FLAG_FULLSCREEN);
//statusBar 显示，navigationBar 不显示
 getWindow().getDecorView().setSystemUiVisibility(
   View.SYSTEM_UI_FLAG_HIDE_NAVIGATION
        | View.SYSTEM_UI_FLAG_VISIBLE);
```

请注意除了添加上述代码，权限也必须添加。

2.16.2 使用系统接口

允许应用隐藏上下的状态栏/导航栏

●应用必须申明 android.permission.CLOUDPOS_HIDE_STATUS_BAR 权限。

●应用调用 PhoneStatusBar 新增的两个方法

1） hideBars 设置 status bar 和 navigation bar 的显示状态。

0-不隐藏全部显示，整个系统有效;1-隐藏 status bar，整个系统有效;2-隐藏 navigation bar，整个系统有效;3-全部隐藏，整个系统有效;

2) getBarsVisibility(返回值同 hideBars 的参数定义),获取 status bar 和 navigation bar 的显示状态。

示例代码如下：

```
//hideBars:
Object service = getSystemService("statusbar");
Class statusBarManager = Class.forName("android.app.StatusBarManager");
 Method method = statusBarManager.getMethod("hideBars", int.class);
 method.invoke(service, 3);
//getBarsVisibility:
Object service = getSystemService("statusbar");
Class statusBarManager = Class.forName("android.app.StatusBarManager");
 Method method = statusBarManager.getMethod("getBarsVisibility");
 Object object = expand.invoke(service);
```

2.17访问权限

2.17.1 权限位置

在使用各接口前，应在 AndroidManifest 文件中声明合适的权限。

2.17.2 权限定义

磁条卡权限

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

接触式 IC 卡权限

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

非接触式 IC 卡权限

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

密码键盘权限

<uses-permission android:name="android.permission.CLOUDPOS_PIN_GET_PIN_BLOCK"/>
<uses-permission android:name="android.permission.CLOUDPOS_PIN_MAC" />
<uses-permission android:name="android.permission.CLOUDPOS_PIN_ENCRYPT_DATA"/>
<uses-permission android:name="android.permission.CLOUDPOS_PIN_UPDATE_USER_KEY"/>
<uses-permission android:name="android.permission. CLOUDPOS_PIN_UPDATE_MASTER_KEY"/>

打印机权限

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

LED 权限

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

串口权限

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

客显权限

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

身份证权限

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

钱箱权限

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

添加 APN 设置权限

<uses-permission android:name=" com.wizarpos.permission.WRITE_APN_SETTINGS "/>

指纹模块权限

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

2.18错误码

2.18.1 简介

错误码是负数，用户可以按照以下方法分析错误码：
把错误码抓成整数：PositiveErrorCode = - ErrorCode
正数错误码:

字节 3	字节 2	字节 1	字节 0
0x00	模块编号	硬件错误码，在各章定义	软件错误码，参见 error.h

模块编号对应的模块:

模块名称	取值	说明
MODULE_ID_PINPAD	1	密码键盘
MODULE_ID_SMARTCARD	2	IC 卡阅读器
MODULE_ID_CONTACTLESS	3	非接卡阅读器

MODULE_ID_ESP	4	外接串口
MODULE_ID_PRINTER	5	打印机
MODULE_ID_MSR	6	磁条卡阅读区
MODULE_ID_HSM	7	安全模块
MODULE_ID_LED	8	LED
MODULE_ID_EEPROM	9	EEPROM
MODULE_ID_IDCARD	10	身份证阅读器