Name	**CLOUDPOS SDK Development Guidance**		
Version	**1.3**	**Number**	
Date	01/18/2016	**State**	[ ]Draft [√]Release [ ]Revising

CLOUDPOS SDK Development Guidance

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	12/16/15	Adjust and optimize layout, add keyboard specification，add APN setting specification.
1.3	Maggie.ma	01/18/2016	Add fingerprint specification

Table of Contents

1 Summary

1.1 Purpose

This document makes standard for system function, application management etc. of application software. Besides, this document makes specification for smart POS devices which are applied to access network.

This document is applicable to the payment terminal used by the merchant to receive the payment, which is not suitable for the payment terminal used by the card holder himself.

1.2 Standard Reference Files

The following files are necessary for this document. Only the date of the reference files which are noted date is compliment for this document. The latest version(including all the modification lists) of the reference files without noting date is compliment for this document

GB/T 2312-1980 Code of Chinese graphic character set for information interchange--Primary set

GB/T 4943.1-2011 Information technology equipment - Safety

GB/T 6833.2～6833.6-1987 Electromagnetic compatibility test specification for electronic measuring instruments

GB/T 9254-2008 Information technology equipment - Radio disturbance characteristics - Limits and methods of measurement

GB/T 14916-1994 Identification cards—Physical characteristics

GB/T 15120.1-.5-1994 Identification cards-Recording technique

GB/T 15694.1-1995 Identification cards—Identification of issuers

GB/T 17552-1998 Identification cards--Financial transaction cards

JR/T 0008-2000 Bank identification number and card number for bank card

JR/T 0025-2013 China Financial IC Card Specifications (PBOC 3.0)

Q/CUP 019-2010 Contactless reader interface specification

Q/CUP 007 UnionPay card acceptance terminal security specification

ISO 7812-2-1993 Identification cards — Identification of issuers

ISO 7816 Identification cards — Integrated circuit cards

ANSI X9.8 Banking - Personal Identification Number Management and Security

1.3 Terms and definitions

1.3.1 Bank Card

A bank card is typically a plastic card issued by a bank to its clients that performs one or more services that relate to giving the client access to funds, either from the client's own bank account, or through a credit account.

1.3.2 Card Holder

Legitimate holder of the bank card who is the customer linked with the bank account of the card.

1.3.3 MSR

A bank card which physical characters is accordance with GB/T 14916 standards and magnetic recording is accordance with GB/T 15120、GB/T 15694-1、ISO 7812-2、GB/T17552 standards.

1.3.4 Integrated Circuit Card

A card embedded one or more integrated circuits to execute processing and storage capabilities.

1.3.5 Personal Identification Number (PIN)

A personal identification number (PIN, pronounced "pin"; often redundantly PIN number) is a numeric password shared between a user and a system, that can be used to authenticate the user to the system.

1.3.6 Sensitive Data (Information)

Sensitive data refers to the track information or unique data of the terminal or the card holder such as PIN and encryption keys. Sensitive data needs to be protected to prevent leakage, to be changed or to be destroyed.

1.3.7 API

In computer programming, an application programming interface (API) is a set of routines, protocols, and tools for building software applications. An API expresses a software component in terms of its operations, inputs, outputs, and underlying types. An API defines functionalities that are independent of their respective implementations, which allows definitions and implementations to vary without compromising the interface.

2 Device API and Specification

2.1 MSR Reader

2.1.1 Drivers

Driver filename is “libUnionpayCloudPos.so” , this so file is integrated in the firmware.

Corresponding interface file of C is “msr_driver_interface.h”

2.1.2 API Interface

open

(1) Function Description

Open the MSR reader device. If success, the device object establishes a connection with the MSR so that it can carry out the following operations.

(2) Interface Format

int msr_open()

(3) Parameter Description

No parameter

(4) Return Value

Return value: >=0, success;
<0, error code.

(5) Error Code Definition

Refer to Error Code and Hardware Error Code.

(6) Direction of Use

This operation should be used before other operations in this chapter.

close

(1) Function Description

Close the opened MSR reader device. If you want to call other APIs, you should open the device again.

(2) Interface Format

int msr_close ()

(3) Parameter Description

No parameter

(4) Return Value

Return value: >=0, success;
<0, error code.

(5) Error Code Definition

Refer to Error Code and Hardware Error Code.

(6) Direction of Use

Open and close are pair operations. If you don't want to use the device, you should call close to release the device.

register_notifier

(1) Function Description

Register a notifier. As soon as data is ready, this notifier will be called.

(2) Interface Format

int msr_register_notifier(MSR_NOTIFIER pNotifier, void* pUserData)

(3) Parameter Description

parameter	type	instruction
pNotifier	MSR_NOTIFIER	Notifier of MSR card reader
pUserData	void*	User data

(4) Return Value

Return value: >=0, success;
<0, error code.

(5) Error Code Definition

Refer to Error Code and Hardware Error Code.

(6) Direction of Use

typedef void (*MSR_NOTIFIER)(void* pUserData)
This operation can be called if you want to monitor the swing card information after opening the device.

unregister_notifier

(1) Function Description

Unregister a notifier.

(2) Interface Format

int msr_unregister_notifier()

(3) Parameter Description

No parameter

(4) Return Value

Return value: >=0, success;
<0, error code.

(5) Error Code Definition

Refer to Error Code and Hardware Error Code.

(6) Direction of Use

If you have register a notifier and you don't want to monitor swing, you can call this api to unregister the notifier.

get_track_error

(1) Function Description

Get error information from the selected track.

(2) Interface Format

int msr_get_track_error(int nTrackIndex)

(3) Parameter Description

parameter	type	instruction
nTrackIndex	int	Track index

(4) Return Value

Return value: >=0, no error;
<0, error code.

(5) Error Code Definition

Refer to Error Code and Hardware Error Code.

(6) Direction of Use

The track index should be 0, 1, 2.
After the notifier found a card, if you want to get information from the selected track of the card, you should firstly call this api. If the result value is more than 0, the following apis can be called.

get_track_data_length

(1) Function Description

Get the length information from the selected track.

(2) Interface Format

int msr_get_track_data_length(int nTrackIndex)

(3) Parameter Description

parameter	type	instruction

nTrackIndex	int	Track index

(4) Return Value

Return value: >=0, length of track data;
<0, error code.

(5) Error Code Definition

Refer to Error Code and Hardware Error Code.

(6) Direction of Use

You can allocate the memory space for the information in the selected track which is used in the following api.

get_track_data

(1) Function Description

Get the data information from the selected track.

(2) Interface Format

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

(3) Parameter Description

parameter	type	instruction
nTrackIndex	int	Track index
pTrackData	unsigned char*	Data buffer
nLength	int	Length of data buffer

(4) Return Value

Return value: >=0, length of track data;
<0, error code.

(5) Error Code Definition

Refer to Error Code and Hardware Error Code.

2.1.3 Hardware Error Code

Error name	value	instruction
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 Call Order

open→register_notifier→get_track_error→get_track_data_length→get_track_data→close

2.2 Smart Card Reader

2.2.1 Drivers

Driver filename is “libUnionpayCloudPos.so” , this so file is integrated in the firmware.

Corresponding interface file of C is “smart_card_interface.h”

2.2.2 API Interface

query_max_number

(1) Function Description

Query the max slot in this smart card reader.

(2) Interface Format

int smart_card_query_max_number()

(3) Parameter Description

No parameter

(4) Return Value

Return value: > 0, number of slot
==0, not defined;
<0, error code.

(5) Error Code Definition

Refer to Error Code and Hardware Error Code.

(6) Direction of Use

Normally the IC card slot index is 0, other slots are used for PSAM card.

query_presence

(1) Function Description

Query whether the smart card is existent in the selected slot.

(2) Interface Format

int smart_card_query_presence(int nSlotIndex)

(3) Parameter Description

parameter	type	instruction

nSlotIndex	int	Slot index, from 0 to MAX - 1

(4) Return Value

Return value: > 0, be existent
==0, not existent;
<0, error code.

(5) Error Code Definition

Refer to Error Code and Hardware Error Code.

(6) Direction of Use

Attention: not every slot can support this function.
This api can be used before open operation.

open

(1) Function Description

Open the specified slot of the smart card reader.

(2) Interface Format

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

(3) Parameter Description

parameter	type	instruction
nSlotIndex	int	Slot index, from 0 to MAX - 1
pNotify	SMART_CARD_NOTIFIER	Notifier of smart card reader
pUserData	void*	User data

(4) Return Value

Return value: >= 0, success, value as handle of this device;
<0, error code.

(5) Error Code Definition

Refer to Error Code and Hardware Error Code.

(6) Direction of Use

A notifier is registered after this api is called.

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) Function Description

Close the specified slot of the smart card reader which is opened before.

(2) Interface Format

int smart_card_close(int nHandle)

(3) Parameter Description

parameter	type	instruction
nHandle	int	Handle of this device, returned from open

(4) Return Value

Return value: >= 0, success;
<0, error code.

(5) Error Code Definition

Refer to Error Code and Hardware Error Code.

(6) Direction of Use

Open and close are pair operations. If you don't want to use the device, you should call close to release the device.

The nSlotIndex of open and close should be all the same.

set_slot_info

(1) Function Description

The function is responsible for writing data to memory card

(2) Interface Format

int smart_card_set_slot_info(int nHandle, SMART_CARD_SLOT_INFO* pSlotInfo)

(3) Parameter Description

parameter	type	instruction
nHandle	int	Handle of this device, returned from open
pSlotInfo	SMART_CARD_SLOT_INFO*	slot information

(4) Return Value

Return value: >= 0, success;
<0, error code.

(5) Error Code Definition

Refer to Error Code and Hardware Error Code.

(6) Direction of Use

```
typedef struct smart_card_slot_info
{
    /** \show initializer Speed communication (parameter TA1 of ISO 7816-3) */
    unsigned char FIDI;

    /** \show initializer Extra Guard Time (parameter TC1 or N of ISO 7816-3) */
    unsigned char EGT;

    /**    \show initializer
     * If protocol T=0 is selected, the parameter indicates the Waiting
     * Integer (parameter TC2 of ISO 7816-3 - the default value is 10)
     * If the protocol T=1 is selected, the parameter indicates the
     * Block and Character Waiting Time Integer (parameter TB3 of ISO 7816-3)
     */
    unsigned char WI;

     /** \show initializer If the protocol T=1 is selected, the parameter indicates the
       * Waiting Time Extention (the default value is 1). */
    unsigned char WTX;
```

```
    /** \show initializer If the protocol T=1 is selected, the parameter indicates the computing
     * mode for EDC : HAL_SCS_EDC_LRC or HAL_SCS_EDC_CRC (The default value
     * is an LRC) */
    unsigned char EDC;

    /** \show initializer The parameter indicates the selected protocol :
     *    HAL_SCS_PROTOCOL_T0
     *    HAL_SCS_PROTOCOL_T1 */
    unsigned char protocol;


    /** \show initializer The power supply value :
     *    HAL_SCS_POWER_1_8V
     *    HAL_SCS_POWER_3V
     *    HAL_SCS_POWER_5V
     */
    unsigned char power;

    /** \show initializer Convention used to transfer byte :
     *    HAL_SCS_CONV_DIRECT
     *    HAL_SCS_CONV_INVERSE */
    unsigned char conv;

    /** \show initializer If the protocol T=1 is selected, the parameter indicates the Information
     * Field Size for the Card (parameter TA3 of ISO 7816-3). */
    unsigned char IFSC;
    unsigned char reserved[3];
    /** \show initializer Possibility to set Character Waiting Time */
    unsigned int cwt;
    /** \show initializer Possibility to set Block Waiting Time */
    unsigned int bwt;
    /*
     * OR of these items:
     *        SMART_CARD_SLOT_INFO_FIDI        (1 << 0)
     *        SMART_CARD_SLOT_INFO_EGT         (1 << 1)
     *        SMART_CARD_SLOT_INFO_WI          (1 << 2)
     *        SMART_CARD_SLOT_INFO_WTX         (1 << 3)
     *        SMART_CARD_SLOT_INFO_EDC         (1 << 4)
     *        SMART_CARD_SLOT_INFO_PROTOCOL    (1 << 5)
     *        SMART_CARD_SLOT_INFO_POWER       (1 << 6)
     *        SMART_CARD_SLOT_INFO_CONV        (1 << 7)
     *        SMART_CARD_SLOT_INFO_IFSC        (1 << 8)
     *        SMART_CARD_SLOT_INFO_CWT         (1 << 9)
     *        SMART_CARD_SLOT_INFO_BWT         (1 << 10)
     *
     */
    unsigned int nSlotInfoItem;
```

}SMART_CARD_SLOT_INFO;

power_on

(1) Function Description

Power on the smart card in the slot opened before.

(2) Interface Format

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

(3) Parameter Description

parameter	type	instruction
nHandle	int	Handle of this device, returned from open
pATR	unsigned char*	ATR from IC card
pATRLength	unsigned int*	Length of ATR
pSlotInfo	SMART_CARD_SLOT_INFO*	slot information

(4) Return Value

Return value: > 0, success;
<=0, error code.

(5) Error Code Definition

Refer to Error Code and Hardware Error Code.

power_off

(1) Function Description

Power off the smart card in the slot opened before.

(2) Interface Format

int smart_card_power_off(int nHandle)

(3) Parameter Description

parameter	type	instruction
nHandle	int	Handle of this device, returned from open

(4) Return Value

Return value: >= 0, success;
<0, error code.

(5) Error Code Definition

Refer to Error Code and Hardware Error Code.

(6) Direction of Use

This api should be used after power_on.
Power_on and power_off are pair operations.

transmit

(1) Function Description

The function sends a command Application Protocol Data Unit(APDU) to a card and retrieve the response APDU, plus the status words SW1 and SW2.

(2) Interface Format

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

(3) Parameter Description

parameter	type	instruction
nHandle	int	Handle of this device, returned from open
pAPDU	unsigned char*	Command of APDU
nAPDULength	unsigned int	Length of command
pResponse	unsigned char*	Card response of APDU command
pResponseLength	unsigned int	Length of response

(4) Return Value

Return value: >= 0, success;
<0, error code.

(5) Error Code Definition

Refer to Error Code and Hardware Error Code.

(6) Direction of Use

This api should be used between power_on and power_off.
If the inserted card is a CPU card, call power_on, transmit, power_off after open.

mc_verify

(1) Function Description

If the inserted card is a memory card, verify its password.

(2) Interface Format

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

(3) Parameter Description

parameter	type	instruction
nHandle	int	Handle of this device, returned from open
pData	unsigned char*	Key data
nDataLength	unsigned int	Length of key

(4) Return Value

Return value: >= 0, success;
<0, error code.

(5) Error Code Definition

Refer to Error Code and Hardware Error Code.

mc_read

(1) Function Description

The function is responsible for reading memory card information.

(2) Interface Format

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

(3) Parameter Description

parameter	type	instruction
nHandle	int	Handle of this device, returned from open
nAreaType	unsigned int	Area type: 0--main memory, 1--protected memory 2--security momory
pDataBuffer	unsigned char*	Data buffer
nDataLength	unsigned int	Data length of expecting reading
cStartAddress	unsigned char	Starting address to read

(4) Return Value

Return value: >= 0, success, data length;
<0, error code.

(5) Error Code Definition

Refer to Error Code and Hardware Error Code.

mc_write

(1) Function Description

The function is responsible for writing data to memory card

(2) Interface Format

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

(3) Parameter Description

parameter	type	instruction
nHandle	int	Handle of this device, returned from open
nAreaType	unsigned int	Area type: 0--main memory, 1--protected memory 2--security momory
pData	unsigned char*	Data buffer
nDataLength	unsigned int	Data length of expecting writing
cStartAddress	unsigned char	Starting address to write

(4) Return Value

Return value: >= 0, success, data length;

<0, error code.

(5) Error Code Definition

Refer to Error Code and Hardware Error Code.

(6) Direction of Use

The mc_read and mc_write APIs are used after mc_verify.
If the inserted card is a memory card, call mc_verify, mc_read, mc_write after open.

2.2.3 Hardware Error Code

Error name	value	instruction
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 Call Order

query_max_number and query_presence are independent of open and close.
If the card inserted is a CPU card, the normally call order is:
open→power_on→transmit→power_off→ close
If the card inserted is a memory card, the normally call order is:
open→ power_on →mc_verify→mc_read/mc_write→ power_off →close

2.3 Contactless IC Card Reader

2.3.1 Drivers

Driver filename is “libUnionpayCloudPos.so” , this so file is integrated in the firmware.
Corresponding interface file of C is “contactless_card_interface.h”

2.3.2 API Interface

open

(1) Function Description

Initialize the contactless card reader

(2) Interface Format

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

(3) Parameter Description

parameter	type	instruction
fNotifier	CONTACTLESS_CARD_NOTIFIER	Notifier of contactless card
pUserData	void*	User data
pErrorCode	int*	Error code if return value is equals to 0

(4) Return Value

Return value: 0, failed;
others, success, value is the handle of contactless card device.

(5) Error Code Definition

Refer to Error Code and Hardware Error Code.

(6) Direction of Use

The open api is usually used with the search_target_begin api.

search_target_begin

(1) Function Description

Start searching the contactless card. If you set the nCardMode is auto, reader will try to activate card in type A, type B and type successively; If you set the nCardMode is type A, type B, or type C, reader only try to activate card in the specified way.

(2) Interface Format

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

(3) Parameter Description

parameter	type	instruction
nHandle	int	Handle of this device, returned from open
nCardMode	int	Mode to search
nFlagSearchAll	int	Not used
nTimeout_MS	int	Time out in millseconds. If it is less than 0, then wait forever.

(4) Return Value

Return value: >= 0, success;
<0, error code.

(5) Error Code Definition

Refer to Error Code and Hardware Error Code.

(6) Direction of Use

Possible value of nCardMode :
#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
You can terminate it using function search_target_end.

search_target_end

(1) Function Description

Stop the process of searching card.

(2) Interface Format

int contactless_card_search_target_end(int nHandle)

(3) Parameter Description

parameter	type	instruction

nHandle	int	Handle of this device, returned from open

(4) Return Value

Return value: >= 0, success;
<0, error code.

(5) Error Code Definition

Refer to Error Code and Hardware Error Code.

(6) Direction of Use

The search_target_begin and search_target_end apis are pair operations.

close

(1) Function Description

Close the contactless card reader.

(2) Interface Format

int contactless_card_close(int nHandle)

(3) Parameter Description

parameter	type	instruction
nHandle	int	Handle of this device, returned from open

(4) Return Value

Return value: >= 0, success;
<0, error code.

(5) Error Code Definition

Refer to 14 Error Code and 2.5 Hardware Error Code

(6) Direction of Use

Open and close are pair operations. If you don’t want to use the device, you should call close to release the device.

query_info

(1) Function Description

Query the number and type of cards on the contactless card reader.

(2) Interface Format

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

(3) Parameter Description

parameter	type	instruction
nHandle	int	Handle of this device, returned from open
pHasMoreCards	int*	Card number
pCardType	int*	Card type

(4) Return Value

Return value: >= 0, success;
<0, error code.

(5) Error Code Definition

Refer to Error Code and Hardware Error Code.

(6) Direction of Use

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) Function Description

Attach the target before transmitting APDU command. In this process, the target(card) is activated and return ATR.

(2) Interface Format

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

(3) Parameter Description

parameter	type	instruction
nHandle	int	Handle of this device, returned from open
pATRBuffer	unsigned char*	ATR buffer, if null, you can not get data.
nATRBufferLength	int	Length of ATR buffer

(4) Return Value

Return value: > 0, success, length of ATR;
<=0, error code.

(5) Error Code Definition

Refer to Error Code and Hardware Error Code.

(6) Direction of Use

If the card is a CPU card, the attach_target, detach_target and transmit apis are used to get informations from the card.

detach_target

(1) Function Description

Detach the target. If you want to send APDU again, you should attach it.

(2) Interface Format

int contactless_card_detach_target(int nHandle)

(3) Parameter Description

parameter	type	instruction
nHandle	int	Handle of this device, returned from open

(4) Return Value

Return value: >= 0, success;
<0, error code.

(5) Error Code Definition

Refer to Error Code and Hardware Error Code.

(6) Direction of Use

Only when you remove the card can you get the return value of this api.

transmit

(1) Function Description

Transmit APDU command and get the response

(2) Interface Format

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

(3) Parameter Description

parameter	type	instruction
nHandle	int	Handle of this device, returned from open
pAPDU	unsigned char*	Command of APDU
nAPDULength	unsigned int	Length of command
pResponse	unsigned char*	Response buffer of APDU command
pResponseLength	unsigned int	[in], buffer length of response [out], length of response

(4) Return Value

Return value: >= 0, success;
<0, error code.

(5) Error Code Definition

Refer to Error Code and Hardware Error Code.

(6) Direction of Use

The APDU command of contactless card is the same as the IC card normally.

verify

(1) Function Description

Verify the key of the contactless card. You should firstly know the key and corresponding key type(key A or key B).

(2) Interface Format

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

(3) Parameter Description

parameter	type	instruction
nHandle	int	Handle of this device, returned from open
nSectorIndex	unsigned int	Sector index
nPinType	unsigned int	Key type, 0—Key A 1—Key B
strPin	unsigned char*	Password of this sector
nPinLength	unsigned int	Length of password

(4) Return Value

Return value: >= 0, success;
<0, error code.

(5) Error Code Definition

Refer to Error Code and Hardware Error Code.

(6) Direction of Use

If the card is a Mifare card, the verify, read and write apis are used to get informations from the card.

read

(1) Function Description

Read data from the contactless card. You should verify the key at first.

(2) Interface Format

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

(3) Parameter Description

parameter	type	instruction
nHandle	int	Handle of this device, returned from open
nSectorIndex	unsigned int	Sector index
nBlockIndex	unsigned int	Block index of the specified sector
pDataBuffer	unsigned char*	Data buffer to read
nDataBufferLength	unsigned int	Buffer length

(4) Return Value

Return value: >= 0, success;
<0, error code.

(5) Error Code Definition

Refer to Error Code and Hardware Error Code.

(6) Direction of Use

You should firstly know the card format(how many sectors, blocks and block size) from the card producer.

write

(1) Function Description

Write data to the contactless card. You should verify the key at first.

(2) Interface Format

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

(3) Parameter Description

parameter	type	instruction
nHandle	int	Handle of this device, returned from open
nSectorIndex	unsigned int	Sector index
nBlockIndex	unsigned int	Block index of the specified sector
pData	unsigned char*	Data buffer to write
nDataLength	unsigned int	Buffer length

(4) Return Value

Return value: >= 0, success;
<0, error code.

(5) Error Code Definition

Refer to Error Code and Hardware Error Code.

write_value

(1) Function Description

Write money value and user data to the specified block if the card is an electronic purse format card.

(2) Interface Format

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

(3) Parameter Description

parameter	type	instruction
nHandle	int	Handle of this device, returned from open
nSectorIndex	unsigned int	Sector index
nBlockIndex	unsigned int	Block index of the specified sector
pValue	unsigned char*	money value to write
nValueBufLength	unsigned int	Buffer length
pAddrData	unsigned char	User data

(4) Return Value

Return value: >= 0, success;
<0, error code.

(5) Error Code Definition

Refer to Error Code and Hardware Error Code.

(6) Direction of Use

This api will format the MC card as an electronic format card. If you want to use the following apis, you should firstly confirm that the card is an electronic purse format card.

read_value

(1) Function Description

Read money value and user data from the specified block if the card is an electronic purse format card.

(2) Interface Format

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

(3) Parameter Description

parameter	type	instruction
nHandle	int	Handle of this device, returned from open
nSectorIndex	unsigned int	Sector index
nBlockIndex	unsigned int	Block index of the specified sector
pValue	unsigned char*	[out]buffer for saving value. LSB, 4 bytes
nValueBufLength	unsigned int	Buffer length
pAddrData	unsigned char*	[out]buffer for saving a user data, 1 byte

(4) Return Value

Return value: >= 0, success;
<0, error code.

(5) Error Code Definition

Refer to Error Code and Hardware Error Code.

increment

(1) Function Description

Increase value to a block. This api should be used with restore and transfer.

(2) Interface Format

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

(3) Parameter Description

parameter	type	instruction
nHandle	int	Handle of this device, returned from open
nSectorIndex	unsigned int	Sector index
nBlockIndex	unsigned int	Block index of the specified sector
pValue	unsigned char*	money value to write
nValueLength	unsigned int	Buffer length, must be greater than 4

(4) Return Value

Return value: >= 0, success;
<0, error code.

(5) Error Code Definition

Refer to Error Code and Hardware Error Code.

decrement

(1) Function Description

Decrease value to a block. This api should be used with restore and transfer.

(2) Interface Format

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

(3) Parameter Description

parameter	type	instruction
nHandle	int	Handle of this device, returned from open

nSectorIndex	unsigned int	Sector index
nBlockIndex	unsigned int	Block index of the specified sector
pValue	unsigned char*	money value to write
nValueLength	unsigned int	Buffer length, must be greater than 4

(4) Return Value

Return value: >= 0, success;
<0, error code.

(5) Error Code Definition

Refer to Error Code and Hardware Error Code.

restore

(1) Function Description

Read the money value to the temporary from a block. This is a necessary step if you want to increase or decrease the value of the electronic purse card.

(2) Interface Format

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

(3) Parameter Description

parameter	type	instruction
nHandle	int	Handle of this device, returned from open
nSectorIndex	unsigned int	Sector index
nBlockIndex	unsigned int	Block index of the specified sector

(4) Return Value

Return value: >= 0, success;
<0, error code.

(5) Error Code Definition

Refer to Error Code and Hardware Error Code.

(6) Direction of Use

This api should be used before increment or decrement.

transfer

(1) Function Description

Save the value to a block from the temporary buffer. This api will really modify the money value of a block.

(2) Interface Format

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

(3) Parameter Description

parameter	type	instruction
nHandle	int	Handle of this device, returned from open
nSectorIndex	unsigned int	Sector index
nBlockIndex	unsigned int	Block index of the specified sector

(4) Return Value

Return value: >= 0, success;
<0, error code.

(5) Error Code Definition

Refer to Error Code and Hardware Error Code.

(6) Direction of Use

This api should be used after increment or decrement.

2.3.3 Hardware Error Code

2.3.4 Call Order

If the card swapped is a CPU card, the normally call order is:
open→attach_target→transmit→detach_target→ close
If the card swapped is a memory card such as Mifare1 card, the normally call order is:
open→ verify→read/write→ close
If the card swapped is an electronic purse memory card, the normally call order is:
open→ verify→read_value/write_value/increment/decrement→ close

The increment or decrement api is usually called like this:
restore→increment/decrement→transfer.

2.4 PINPad

2.4.1 Drivers

Driver filename is “libUnionpayCloudPos.so” , this so file is integrated in the firmware.

Corresponding interface file of C is “pinpad_interface.h”

2.4.2 API Interface

open

(1) Function Description

Open the PINPad device.

(2) Interface Format

int pinpad_open()

(3) Parameter Description

No parameter

(4) Return Value

Return value: >= 0, success;
<0, error code.

(5) Error Code Definition

Refer to Error Code and Hardware Error Code

(6) Direction of Use

This operation should be used before other operations in this chapter.

close

(1) Function Description

Close the PINPad device.

(2) Interface Format

int pinpad_close()

(3) Parameter Description

No parameter

(4) Return Value

Return value: >= 0, success;
<0, error code.

(5) Error Code Definition

Refer to Error Code and Hardware Error Code

(6) Direction of Use

This operation should be used if you don't want to use this device.

show_text

(1) Function Description

Show text in the specified line.

(2) Interface Format

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

(3) Parameter Description

parameter	type	instruction
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) Return Value

Return value: >= 0, success;
<0, error code , maybe your display string is too long.

(5) Error Code Definition

Refer to Error Code and Hardware Error Code

(6) Direction of Use

If the terminal device is a WIZARHAND_Q1, the text just can be String.getBytes(). If the device is external PINPad, the text of Chinese characters see attachment: UserGuide-PinpadShowCharset.txt.

update_user_key

(1) Function Description

Update the user key. You should check the cipher user key by yourself through the specified master key and check value obtained from the server or other.

(2) Interface Format

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

(3) Parameter Description

parameter	type	instruction
nMasterKeyID	int	Master key id
nUserKeyID	int	User key id
pCipherNewUserKey	unsigned char*	New user key in cipher text
nCipherNewUserKeyLength	int	Length of new user key

(4) Return Value

Return value: >= 0, success;
<0, error code.

(5) Error Code Definition

Refer to Error Code and Hardware Error Code

(6) Direction of Use

If the device is external PINPad, you should update the user key by this api.

update_user_key_with_check

(1) Function Description

Update the user key with check value. You don't need to check the cipher user key by yourself.

(2) Interface Format

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

(3) Parameter Description

parameter	type	instruction
nMasterKeyID	int	Master key id
nUserKeyID	int	User key id
pCipherNewUserKey	unsigned char*	New user key in cipher text
nCipherNewUserKeyLength	int	Length of new user key
nKeyUsge	int	Key type. 0--PIN key; 1--MAC key; 2—Data key
pCheckValue	unsigned char*	Check value of user key
nCheckValueLen	int	Length of check value, 4 bytes in general

(4) Return Value

Return value: >= 0, success;
<0, error code.

(5) Error Code Definition

Refer to Error Code and Hardware Error Code

(6) Direction of Use

If the terminal device is a WIZARHAND_Q1, this api should be used to update the user key.

set_pin_length

(1) Function Description

Set the max or min length of PIN. When you call the calculate_pin_block api, the number you can input is no more than the max length.

(2) Interface Format

int pinpad_set_pin_length(int nLength, int nFlag)

(3) Parameter Description

parameter	type	instruction
nLength	int	PIN length
nFlag	int	Flag, 0--min length 1--max length

(4) Return Value

Return value: >= 0, success;
<0, error code.

(5) Error Code Definition

Refer to Error Code and Hardware Error Code

(6) Direction of Use

This api should be used before calculate_pin_block if you want to limit the length of PIN.

get_serial_number

(1) Function Description

Get serial number from the PINPad.

(2) Interface Format

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

(3) Parameter Description

parameter	type	instruction
pData	unsigned char*	Serial number buffer
nBufferLength	unsigned int	Length of buffer

(4) Return Value

Return value: >= 0, success, length of serial number;
<0, error code.

(5) Error Code Definition

Refer to Error Code and Hardware Error Code

select_key

(1) Function Description

Select master key and user key before encryption operations.

(2) Interface Format

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

(3) Parameter Description

parameter	type	instruction
nKeyType	int	1 : TDUKPT, 2 : MASTER-SESSION PAIR
nMasterKeyID	int	Master key id, 0-9 when nKeyType is master-session; Master key id, 0,1,2when nKeyType is TDUKPT
nUserKeyID	int	User key id, used when nKeyType is master-session.
nAlgorith	int	Not used

(4) Return Value

Return value: >= 0, success;
<0, error code.

(5) Error Code Definition

Refer to Error Code and Hardware Error Code

(6) Direction of Use

If the terminal device is a WIZARHAND_Q1, the user key id is 0-2 when nKeyType is master-session pair. If the device is external PINPad, the user key id is 0-1 when nKeyType is master-session pair.

calculate_pin_block

(1) Function Description

Calculate the PIN block of the inputted PIN. You should call select_key at first. The user key should be a PIN key whose id is 0 in general.

(2) Interface Format

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

(3) Parameter Description

parameter	type	instruction
pASCIICardNumber	unsigned char*	Card number in ASCII format
nCardNumberLength	int	Length of card number
pPinBlockBuffer	unsigned char*	[out]buffer for saving PIN block
nPinBlockBufferLength	int	Length of buffer
nTimeout_MS	int	Timeout waiting for user input in milliseconds. If it is less than 0, then wait forever.
nFlagSound	int	Not used

(4) Return Value

Return value: >= 0, success, length of pin block;
<0, error code.

(5) Error Code Definition

Refer to Error Code and Hardware Error Code

(6) Direction of Use

The length of card number is normally 13-19.
The max waiting time for PIN input is 60s, and the two PIN input time span is 10s, otherwise this API will return error code.

calculate_mac

(1) Function Description

Calculate the MAC. The user key should be a MAC key whose id is 1 in general.

(2) Interface Format

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

(3) Parameter Description

parameter	type	instruction
pData	unsigned char*	data
nDataLength	int	Data length
nMACFlag	int	0:X99, 1: ECB,2:SE919,3:UnionPayECB
pMACOutBuffer	unsigned char*	[out]MAC data buffer
nMACOutBufferLength	int	Length of MAC data buffer

(4) Return Value

Return value: >= 0, success, length of pin block;
<0, error code.

(5) Error Code Definition

Refer to Error Code and Hardware Error Code

encrypt_string

(1) Function Description

Encrypt string. The user key should be a data key whose id is 2 in general.

(2) Interface Format

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

(3) Parameter Description

parameter	type	instruction
pPlainText	unsigned char*	Plain text

nTextLength	int	Length of plain text
pCipherTextBuffer	unsigned char*	[out]buffer for saving cipher text
nCipherTextBufferLength	int	Length of buffer

(4) Return Value

Return value: >= 0, success, length of pin block;
<0, error code.

(5) Error Code Definition

Refer to Error Code and Hardware Error Code

(6) Direction of Use

The default Mode is ECB/NoPadding.

update_cipher_master_key

(1) Function Description

Update master key, the master key must be ciphered by transport key.

(2) Interface Format

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

(3) Parameter Description

parameter	type	instruction
nMasterKeyID	int	Master key index
pCipherMasterKey	unsigned char*	Ciphered master key
nCipherMasterKeyLen	int	Length of ciphered master key
pCheckValue	unsigned char*	Check value
nCheckValueLen	int	Length of check value

(4) Return Value

Return value: >= 0, success;
<0, error code.

(5) Error Code Definition

Refer to Error Code and Hardware Error Code

update_user_key_with_check_E

(1) Function Description

Update the user key with check value. You can set the the algorithm of check.

(2) Interface Format

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) Parameter Description

parameter	type	instruction
nMasterKeyID	int	Master key id
nUserKeyID	int	User key id
pCipherNewUserKey	unsigned char*	New user key in cipher text
nCipherNewUserKeyLength	int	Length of new user key
nKeyUsge	int	Key type. 0--PIN key; 1--MAC key; 2—Data key
pCheckValue	unsigned char*	Check value of user key
nCheckValueLen	int	Length of check value, 4 bytes in general
algoCheckValue	int	ALGO_CHECK_VALUE: 0-- ALGO_CHECK_VALUE_DEFAULT 1-- ALGO_CHECK_VALUE_SE919

(4) Return Value

Return value: >= 0, success;
<0, error code.

(5) Error Code Definition

Refer to Error Code and Hardware Error Code

update_cipher_master_key_E

(1) Function Description

Update master key, the master key must be ciphered by transport key, and use the selected algorithm of check .

(2) Interface Format

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

(3) Parameter Description

parameter	type	instruction
nMasterKeyID	int	Master key index
pCipherMasterKey	unsigned char*	Ciphered master key
nCipherMasterKeyLen	int	Length of ciphered master key
pCheckValue	unsigned char*	Check value
nCheckValueLen	int	Length of check value
algoCheckValue	int	ALGO_CHECK_VALUE: 0-- ALGO_CHECK_VALUE_DEFAULT 1-- ALGO_CHECK_VALUE_SE919

(4) Return Value

Return value: >= 0, success;
<0, error code.

(5) Error Code Definition

Refer to Error Code and Hardware Error Code

encrypt_string_with_mode

(1) Function Description

Encrypt string.

(2) Interface Format

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) Parameter Description

parameter	type	instruction
pPlainText	unsigned char*	Plain text
nTextLength	int	Length of plain text
pCipherTextBuffer	unsigned char*	[out]buffer for saving cipher text, for dukpt encrypt, the buffer data structure is: cipher data + KSN + counter.
nCipherTextBufferLength	int	Length of buffer
nMode	int	Cipher mode: 0--PINPAD_ENCRYPT_STRING_MODE_EBC 1--PINPAD_ENCRYPT_STRING_MODE_CBC 2--PINPAD_ENCRYPT_STRING_MODE_CFB 3--PINPAD_ENCRYPT_STRING_MODE_OFB Use NoPadding.
pIV	unsigned char*	initial vector, only for CBC, CFB, OFB mode
nIVLen	int	length of IV, must be equal to block length according to the algorithm

(4) Return Value

Return value: >= 0, success; the value is the length of the cipher data length.
<0, error code.

(5) Error Code Definition

Refer to Error Code and Hardware Error Code

2.4.3 Hardware Error Code

Error name	value	instruction
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 Call Order

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

2.5 Printer

2.5.1 Drivers

Driver filename is “libUnionpayCloudPos.so” , this so file is integrated in the firmware.

Corresponding interface file of C is “printer_interface.h”

2.5.2 API Interface

open

(1) Function Description

Open the printer device.

(2) Interface Format

int printer_open()

(3) Parameter Description

No parameter

(4) Return Value

Return value: >= 0, success;
<0, error code.

(5) Error Code Definition

Refer to Error Code and Hardware Error Code.

(6) Direction of Use

This operation should be used before other operations. The open api is usually used with the begin api.

begin

(1) Function Description

Prepare to print

(2) Interface Format

int printer_begin()

(3) Parameter Description

No parameter

(4) Return Value

Return value: >= 0, success;
<0, error code.

(5) Error Code Definition

Refer to Error Code and Hardware Error Code.

(6) Direction of Use

You can print data after this api.
You can terminate it using the end api.

end

(1) Function Description

End to print

(2) Interface Format

int printer_end()

(3) Parameter Description

No parameter

(4) Return Value

Return value: >= 0, success;
<0, error code.

(5) Error Code Definition

Refer to Error Code and Hardware Error Code.

(6) Direction of Use

The begin and end apis are pair operations.

close

(1) Function Description

Close the device.

(2) Interface Format

int printer_close()

(3) Parameter Description

No parameter

(4) Return Value

Return value: >= 0, success;
<0, error code.

(5) Error Code Definition

Refer to Error Code and Hardware Error Code.

(6) Direction of Use

The open and close apis are pair operations. If you don't want to use the device, you should call close to release the device.

query_status

(1) Function Description

Query the status of the printer.

(2) Interface Format

int printer_query_status()

(3) Parameter Description

No parameter

(4) Return Value

Return value: == 1, has paper.
== 0, success;
< 0, error code.

(5) Error Code Definition

Refer to Error Code and Hardware Error Code.

(6) Direction of Use

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

write

(1) Function Description

Write data to the device. The data can be String or Bitmap data.

(2) Interface Format

int printer_write(unsigned char* pData, int nDataLength)

(3) Parameter Description

parameter	type	instruction
pData	unsigned char*	Data or control command
nDataLength	int	Length of data

(4) Return Value

Return value: == 1, has paper.
== 0, success;
< 0, error code.

(5) Error Code Definition

Refer to Error Code and Hardware Error Code.

2.5.3 Hardware Error Code

2.5.4 Call Order

open→query_status→begin→write→end→ close

2.6 LED

2.6.1 Drivers

Driver filename is “libUnionpayCloudPos.so” , this so file is integrated in the firmware.

Corresponding interface file of C is “led_service_interface.h”

2.6.2 API Interface

open

(1) Function Description

Open the LED device, which contains all the LEDs' service.

(2) Interface Format

int led_open()

(3) Parameter Description

No parameter

(4) Return Value

Return value: >= 0, success;
<0, error code.

(5) Error Code Definition

Refer to Error Code and Hardware Error Code.

(6) Direction of Use

This operation should be used before other operations.

close

(1) Function Description

Close the LED device, which contains all the LEDs' service.

(2) Interface Format

int led_close()

(3) Parameter Description

No parameter

(4) Return Value

Return value: >= 0, success;
<0, error code.

(5) Error Code Definition

Refer to Error Code and Hardware Error Code.

(6) Direction of Use

The open and close apis are pair operations. If you don't want to use this device, you should call the close api to release this device.

turn_on

(1) Function Description

Turn on the specified LED.

(2) Interface Format

int led_on(unsigned int nLedIndex)

(3) Parameter Description

parameter	type	instruction
nLedIndex	unsigned int	Index of led, >= 0 && < MAX_LED_COUNT

(4) Return Value

Return value: >= 0, success;
<0, error code.

(5) Error Code Definition

Refer to Error Code and Hardware Error Code.

turn_off

(1) Function Description

Turn off the specified LED.

(2) Interface Format

int led_off(unsigned int nLedIndex)

(3) Parameter Description

parameter	type	instruction
nLedIndex	unsigned int	Index of led, >= 0 && < MAX_LED_COUNT

(4) Return Value

Return value: >= 0, success;
<0, error code.

(5) Error Code Definition

Refer to Error Code and Hardware Error Code.

get_status

(1) Function Description

Get the status of the specified LED.

(2) Interface Format

int led_get_status(unsigned int nLedIndex)

(3) Parameter Description

parameter	type	instruction
nLedIndex	unsigned int	Index of led, >= 0 && < MAX_LED_COUNT

(4) Return Value

Return value: > 0, LED on;
= 0, LED off;
< 0, error code.

(5) Error Code Definition

Refer to Error Code and Hardware Error Code.

2.6.3 Hardware Error Code

2.6.4 Call Order

open→turn_on/turn_off/get_status → close

2.7 Serial Port

2.7.1 Drivers

Driver filename is "libUnionpayCloudPos.so" , this so file is integrated in the firmware.

Corresponding interface file of C is "ext_serial_port_interface.h"

2.7.2 API Interface

open

(1) Function Description

Open the serial port by the specified device name.

(2) Interface Format

int esp_open(char* pDeviceName)

(3) Parameter Description

parameter	type	instruction
pDeviceName	char*	Device name

(4) Return Value

Return value: >= 0, handle of this device;
<0, error code.

(5) Error Code Definition

Refer to Error Code and Hardware Error Code.

(6) Direction of Use

This operation should be used before other operations.
The default pDeviceName of CLOUDPOS is "/dev/s3c2410_serial2".

close

(1) Function Description

Close the serial port opened before.

(2) Interface Format

int esp_close(int nHandle)

(3) Parameter Description

parameter	type	instruction
nHandle	int	Handle of this device, returned from open

(4) Return Value

Return value: >= 0, success;
<0, error code.

(5) Error Code Definition

Refer to Error Code and Hardware Error Code.

(6) Direction of Use

The open and close apis are pair operations. If you don't want to use this device, you should call the close api to release this device.

set_baudrate

(1) Function Description

Set the baud rate of the serial port so that this device can read and write in the same baud rate.

(2) Interface Format

int esp_set_baudrate(int nHandle, unsigned int nBaudrate)

(3) Parameter Description

parameter	type	instruction
nHandle	int	Handle of this device, returned from open
nBaudrate	int	Baud rate

(4) Return Value

Return value: >= 0, success;
<0, error code.

(5) Error Code Definition

Refer to Error Code and Hardware Error Code.

(6) Direction of Use

This api should be used before read and write.

read

(1) Function Description

Get information from the serial port.

(2) Interface Format

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

(3) Parameter Description

parameter	type	instruction
nHandle	int	Handle of this device, returned from open
pDataBuffer	unsigned char*	Data buffer
nExpectedDataLength	int	Data length to read
nTimeout_MS	int	Time in milliseconds. 0 : read and return immediately <0: read until got data.

(4) Return Value

Return value: > 0, data length;
<= 0, error code.

(5) Error Code Definition

Refer to Error Code and Hardware Error Code.

write

(1) Function Description

Send information from the serial port.

(2) Interface Format

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

(3) Parameter Description

parameter	type	instruction
nHandle	int	Handle of this device, returned from open
pDataBuffer	unsigned char*	Data buffer
nDataLength	int	Data length

(4) Return Value

Return value: >= 0, written data length;
<0, error code.

(5) Error Code Definition

Refer to Error Code and Hardware Error Code.

flush_io

(1) Function Description

Flush the IO buffer of the serial port.

(2) Interface Format

int esp_flush_io(int nHandle)

(3) Parameter Description

parameter	type	instruction
nHandle	int	Handle of this device, returned from open

(4) Return Value

Return value: >= 0, success;
<0, error code.

(5) Error Code Definition

Refer to Error Code and Hardware Error Code.

setEnable

(1) Function Description

Enable or disable the serial port device.

(2) Interface Format

int esp_setEnable(unsigned int nEnable)

(3) Parameter Description

parameter	type	instruction
nEnable	unsigned int	1 : enable, 0 : disable.

(4) Return Value

Return value: >= 0, success;
<0, error code.

(5) Error Code Definition

Refer to Error Code and Hardware Error Code.

2.7.3 Hardware Error Code

2.7.4 Call Order

open→set_baudrate→read/write→fulsh_io→ close

2.8 CustomerDisplay

2.8.1 Drivers

Driver filename is “libUnionpayCloudPos.so” , this so file is integrated in the firmware.

Corresponding interface file of C is “customer_display_interface.h”

2.8.2 API Interface

open

(1) Function Description

Open the CustomerDisplay device.

(2) Interface Format

void* customer_display_open(int* pError)

(3) Parameter Description

parameter	type	instruction
pErrorCode	int*	Error code if return value is equals to 0

(4) Return Value

Return value: 0, failed;
others, success, value is the handle of contactless card device.

(5) Error Code Definition

Refer to Error Code and Hardware Error Code.

(6) Direction of Use

This operation should be used before other operations.

close

(1) Function Description

Close the CustomerDisplay device opened before.

(2) Interface Format

int customer_display_close(int nHandle)

(3) Parameter Description

parameter	type	instruction
nHandle	int	Handle of this device, returned from open

(4) Return Value

Return value: >= 0, success;
<0, error code.

(5) Error Code Definition

Refer to Error Code and Hardware Error Code.

(6) Direction of Use

The open and close apis are pair operations. If you don't want to use this device, you should call the close api to release this device.

ctrl_devs

(1) Function Description

Use the function of the CustomerDisplay, such as set background, display default screen, open or close led, buzzer beep and so on.

(2) Interface Format

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

(3) Parameter Description

parameter	type	instruction
nHandle	int	Handle of this device, returned from open

nCmd	int	Control command
nValue	int	Value of control command

(4) Return Value

Return value: >= 0, success;
<0, error code.

(5) Error Code Definition

Refer to Error Code and Hardware Error Code.

(6) Direction of Use

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

write_picture

(1) Function Description

Write picture point data(one point to 4 bytes in ARGB8888).

(2) Interface Format

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) Parameter Description

parameter	type	instruction
nHandle	int	Handle of this device, returned from open
nXcoordinate	unsigned int	X cordinate
nYcoordinate	unsigned int	Y cordinate
nWidth	unsigned int	width
nHeight	unsigned int	Height
pData	unsigned char*	All point data
nDataLength	unsigned int	Point data length

(4) Return Value

Return value: >= 0, success;
<0, error code.

(5) Error Code Definition

Refer to Error Code and Hardware Error Code.

(6) Direction of Use

You need firstly convert a Bitmap to point data.

2.8.3 Hardware Error Code

2.8.4 Call Order

open→ctrl_devs/write_picture→ close

2.9 IDCard

2.9.1 Drivers

Driver filename is "libUnionpayCloudPos.so" , this so file is integrated in the firmware.

Corresponding interface file of C is "customer_display_interface.h"

2.9.2 API Interface

open

(1) Function Description

Open the IDCard reader device.

(2) Interface Format

int identity_card_open()

(3) Parameter Description

No parameter

(4) Return Value

Return value: >= 0, handle of this device;
<0, error code.

(5) Error Code Definition

Refer to Error Code and Hardware Error Code.

(6) Direction of Use

This operation should be used before other operations.

close

(1) Function Description

Close the IDCard reader device opened before.

(2) Interface Format

int identity_card_close(int nHandle);

(3) Parameter Description

parameter	type	instruction
nHandle	int	Handle of this device, returned from open

(4) Return Value

Return value: >= 0, success;
<0, error code.

(5) Error Code Definition

Refer to Error Code and Hardware Error Code.

(6) Direction of Use

The open and close apis are pair operations. If you don't want to use this device, you

should call the close api to release this device.

search_target

(1) Function Description

Begin to search IDCard. This api will return until find a IDCard.

(2) Interface Format

int identity_card_search_target(int nHandle)

(3) Parameter Description

parameter	type	instruction
nHandle	int	Handle of this device, returned from open

(4) Return Value

Return value: 0, failed;
others, error code.

(5) Error Code Definition

Refer to Error Code and Hardware Error Code.

get_information

(1) Function Description

Get fixed information of the IDCard.

(2) Interface Format

int identity_card_get_fixed_information(int nHandle, struct tagIDCardData *pIDCardData)

(3) Parameter Description

parameter	type	instruction
nHandle	int	Handle of this device, returned from open
pIDCardData	struct tagIDCardData *	IDCard data

(4) Return Value

Return value: >= 0, success, return picture data length;
<0, error code.

(5) Error Code Definition

Refer to Error Code and Hardware Error Code.

(6) Direction of Use

```
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 Hardware Error Code

2.9.4 Call Order

open→search_target→get_information→ close

2.10 CashDrawer

2.10.1Drivers

Driver filename is “libUnionpayCloudPos.so” , this so file is integrated in the firmware.

Corresponding interface file of C is “moneybox_interface.h”

2.10.2 API Interface

open

(1) Function Description

Open the cash drawer device.

(2) Interface Format

int moneybox_open()

(3) Parameter Description

No parameter

(4) Return Value

Return value: >= 0, success;
<0, error code.

(5) Error Code Definition

Refer to Error Code and Hardware Error Code.

(6) Direction of Use

This operation should be used before other operations.

close

(1) Function Description

Close the cash drawer device.

(2) Interface Format

int moneybox_close()

(3) Parameter Description

No parameter

(4) Return Value

Return value: >= 0, success;
<0, error code.

(5) Error Code Definition

Refer to Error Code and Hardware Error Code.

(6) Direction of Use

The open and close apis are pair operations. If you don't want to use this device, you should call the close api to release this device.

kick_out

(1) Function Description

Kick out the cash drawer.

(2) Interface Format

int moneybox_ctrl()

(3) Parameter Description

No parameter

(4) Return Value

Return value: >= 0, success;
<0, error code.

(5) Error Code Definition

Refer to Error Code and Hardware Error Code.

setEnable

(1) Function Description

Enable or disable hardware

(2) Interface Format

int moneybox_setEnable(unsigned int nEnable)

(3) Parameter Description

parameter	type	instruction
nEnable	unsigned int	1 : enable, 0 : disable.

(4) Return Value

Return value: >= 0, success;
<0, error code.

(5) Error Code Definition

Refer to Error Code and Hardware Error Code.

2.10.3 Hardware Error Code

2.10.4 Call Order

open→kick_out→ close

2.11 KeyBoard

2.11.1 Special Key in Keyboard

There are several special keys in CLOUDPOS series products, all the values of these keys are custom.

Q1 or M0

Key	Key value	description
Period	KeyEvent.KEYCODE_PERIOD	The black dot key on the right-bottom of keyboard.
QRCode	232	The orange key on the middle-bottom of keyboard
Back	KeyEvent.KEYCODE_ESCAPE	The red X key on the left-bottom of keyboard

Scanner_Left	229	The orange key on the left of the LED
Scanner_Right	230	The orange key on the right of the LED ,Only for M0。
Talk	231	Below the scanner left key,only for M0。
Delete	KeyEvent.KEYCODE_DEL	The yellow triangle key on the right-top of keyboard
Enter	KeyEvent.KEYCODE_ENTER	The green circle key on the right-bottom of keyboard

The others are defined in android system. see android.view.KeyEvent.

Pad or POS

All the keys are defined in android system. See android.view.KeyEvent.

2.11.2 Scanner Key or QR Code Key

Use Specification

The Scanner key and the QR Code key can start the scanner application quickly. If one of the keys is pressed, the system will search all the applications installed in the system, the applications with the intent-filter of scanner will be started. If there are many applications, the system will provide a list for the user to select, the selected will be start, The contents in the intent-filter of scanner are as follows:

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

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

Press the Scanner key or the QR Code key will not effect when none of the applications in the system defines the intent-filter of scanner.

Sample

There are two methods to use the Scanner key and the QR Code key:

1. To start scanner application quickly. The application must define the intent-filter of scanner in its' android manifest file, the category and the action definition are as below:

```
<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. Use like the other keys, when press the key , the activated and top task will get the key value and do something. Firstly the application must override the function of

onKeydown, as follows:

```
/** 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) {
        // listen the key code
        if (keyCode == KEYCODE_SCAN_LEFT|| keyCode ==KEYCODE_SCAN_RIGHT|| keyCode ==KEYCODE_QR) {
                //do something, for example:scanner……
                return true;
        }
        return super.onKeyDown(keyCode, event);
}
```

2.12 Fingerprint

2.12.1 Drivers

Driver filename is “libUnionpayCloudPos.so” , this so file is integrated in the firmware.

Corresponding interface file of C is “fingerprint_interface.h”

2.12.2 API Interface

open

(1) Function Description

Open the cash fingerprint.

(2) Interface Format

int fp_open()

(3) Parameter Description

No parameter

(4) Return Value

Return value: >= 0, success;
<0, error code.

(5) Error Code Definition

Refer to Error Code and Hardware Error Code.

(6) Direction of Use

This operation should be used before other operations.

Close

(1) Function Description

Close the fingerprint device.

(2) Interface Format

int fp_close()

(3) Parameter Description

No parameter

(4) Return Value

Return value: >= 0, success;
<0, error code.

(5) Error Code Definition

Refer to Error Code and Hardware Error Code.

(6) Direction of Use

The open and close apis are pair operations. If you don't want to use this device, you should call the close api to release this device.

get_fea

(1) Function Description

Get the feature of fingerprint.

(2) Interface Format

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

(3) Parameter Description

parameter	type	instruction
pFeaBuffer	char *	The buffer to store the feature, not null
nFeaLength	int	The length of the buffer
pRealFeaLength	int *	Reture the real length of the buffer
nTimeOut_S	int	Timeout, unit of time:s

(4) Return Value

Return value: >= 0, success;
<0, error code , maybe your display string is too long.

(5) Error Code Definition

Refer to Error Code and Hardware Error Code

(6) Direction of Use

When the finger touch the fingerprint device, the feature of the fingerprint will be return.

getLastImage

(1) Function Description

Get the image of fingerprint.

(2) Interface Format

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

(3) Parameter Description

parameter	type	instruction
pImgBuffer	char *	The image buffer
nImgLength	int	The length of image buffer
pRealImaLength	int *	The real length of the image buffer

pImgWidth	int *	The width of the image
pImgHeight	int *	The height of the image

(4) Return Value

Return value: >= 0, success;
<0, error code , maybe your display string is too long.

(5) Error Code Definition

Refer to Error Code and Hardware Error Code

(6) Direction of Use

When the finger touch the fingerprint device, the image of the fingerprint will be return.

match

(1) Function Description

Match the fingerprint.

(2) Interface Format

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

(3) Parameter Description

parameter	type	instruction
pFeaBuffer1	char *	The feature of the old fingerprint
nFea1Length	int	The length of the feature
pFeaBuffer2	char *	The feature of the new fingerprint
nFea2Lenghth	int	The length of the feature

(4) Return Value

Return value: > 0, success, the percentage similarity;
<=0, error code , maybe your display string is too long.

(5) Error Code Definition

Refer to Error Code and Hardware Error Code

(6) Direction of Use

Input the two feature of the fingerprint, it will get the result of match .

cancel

(1) Function Description

Cancel the fingerprint device operation.

(2) Interface Format

int fp_cancel ()

(3) Parameter Description

No parameter

(4) Return Value

Return value: >= 0, success;
<0, error code.

(5) Error Code Definition

Refer to Error Code and Hardware Error Code.

2.13 Add APN Setting

2.13.1 API Interface

addByAllArgs

(1) Function Description

Add an APN setting.

(2) Interface Format

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) Parameter Description

parameter	type	instruction
name	String	Not null
apn	String	Not null
mcc	String	Not null
mnc	String	Not null
proxy	String	
port	String	
MMSProxy	String	
MMSPort	String	
userName	String	
server	String	
password	String	
MMSC	String	
authType	String	Null(default),PAP,CHAP,PAP/CHAP
protocol	String	IPV4(default),IPV6,IPV4/IPV6
roamingProtocol	String	IPV4(default),IPV6,IPV4/IPV6
type	String	
bearer	String	Null(default),LTE,eHRPD
MVNOType	String	Null(default),SPN,IMSI,GID
MVNOMatchData	String	

(4) Return Value

Return value: “succeed”,success;
“error description”,fail.

add

(1) Function Description

Add an APN setting.

(2) Interface Format

String add(String name, String apn)

(3) Parameter Description

parameter	type	instruction
name	String	Not null
apn	String	Not null

(4) Return Value

Return value: "succeed",success;
"error description",fail.

addByMCCAndMNC

(1) Function Description

Add an APN setting.

(2) Interface Format

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

(3) Parameter Description

parameter	type	instruction
name	String	Not null
apn	String	Not null
mcc	String	Not null
mnc	String	Not null

(4) Return Value

Return value: "succeed",success;
"error description",fail.

setSelected

(1) Function Description

Set default APN.

(2) Interface Format

boolean setSelected(String name)

(3) Parameter Description

parameter	type	instruction
name	String	名称,必传

(4) Return Value

Return value: true, success;
false, fail.

clear

(1) Function Description

Clear all APN settings.

(2) Interface Format

boolean clear()

(3) Parameter Description

No parameter.

(4) Return Value

Return value: true, success;
false, fail.

2.13.2 Use Specification

The APN interface is com.wizarpos.wizarviewagentassistant.APNManagerService which is provided by the system. The application must import the package when using the interface and add the uses-permission in android manifest file ,see 14.2.11.

2.14 Disable home key

2.14.1 Disable home key in apk

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

Use the permission, the apk will catch the event when the home key or back key has been pressed.

2.14.2 Disable home key in activity

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

Use the permission, and set the activity window type is TYPE_KEYGUARD or TYPE_KEYGUARD_DIALOG, will catch the event when the home key or back key has been pressed.

For example:

```
public void onAttachedToWindow() {
    super.onAttachedToWindow();
    try {

this.getWindow().setType(WindowManager.LayoutParams.TYPE_KEYGUARD);
    } catch (Throwable e) {
        e.printStackTrace();
    }

}
```

This method is used in wizarpos 1, but in WIZARHAND Q1, the app will use SystemUiFlags.Immersive to implement the fuction.

2.15 Full Screen in Q1

2.15.1 Android

1. Grant permission: android.permission.CLOUDPOS_REAL_FULLSCREEN to the app
2. Use immersive mode which hides a device's status bar and navigation bar so that app content is given as much canvas as possible

For example:

```
//statusBar，navigationBar are all not display
getWindow().getDecorView().setSystemUiVisibility(
    View.SYSTEM_UI_FLAG_HIDE_NAVIGATION
    | View.SYSTEM_UI_FLAG_FULLSCREEN);

//statusBar is display, navigationBar is not display
getWindow().getDecorView().setSystemUiVisibility(
    View.SYSTEM_UI_FLAG_HIDE_NAVIGATION
    | View.SYSTEM_UI_FLAG_VISIBLE);
```

2.15.2 System Interface

This will effect in all the system.

1. Grant permission:android.permission.CLOUDPOS_HIDE_STATUS_BAR to the app.
2. Calling the follows method of PhoneStatusBar:

1) hideBars(int) set status bar and navigation bar state

Setting to 1 will hide status bar,2 will hide navigation bar,3 will hide both,0 will show both.

2) getBarsVisibility() get state of status bar and navigation bar.

For Example:

```
//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);
```

2.16 Access permissions

2.16.1 Permission position

All the application should declare proper permission in its Android Manifest file before using each interface.

2.16.2 Permission definition

MSR permission

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

Smart Card permission

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

Contactless Card permission

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

PINPad permission

```
<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_UPDATA_USER_KEY"/>
<uses-permission android:name="android.permission. CLOUDPOS_PIN_UPDATE_MASTER_KEY"/>
```

Printer permission

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

LED permission

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

Serial Port permission

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

CustomerDisplay permission

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

IDCard permission

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

CashDrawer permission

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

Add APN setting permission

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

Fingerprint permission

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

2.17 Error Code

2.17.1Instruction

Error Code is negative number, user can analyze this code by the following method:
Transfer error code to positive value: PositiveErrorCode = - ErrorCode
Positive Error Code:

Byte 3	Byte 2	Byte 1	Byte 0
0x00	Module number	Hardware Error Code, defined in each interface.	Software Error Code, defined in error.h

Module number to module:

Module name	value	instruction
MODULE_ID_PINPAD	1	PINPad
MODULE_ID_SMARTCARD	2	Smart card reader

MODULE_ID_CONTACTLESS	3	Contactless card reader
MODULE_ID_ESP	4	External serial port
MODULE_ID_PRINTER	5	Printer
MODULE_ID_MSR	6	MSR reader
MODULE_ID_HSM	7	HSM
MODULE_ID_LED	8	LED
MODULE_ID_EEPROM	9	EEPROM
MODULE_ID_IDCARD	10	Identity card reader