WIZARPOS International Co., Ltd.

EMV KernelInterface

version 4.05

control info

version	Date	Description	who
1.00	2013-12-20	create	Michael Li
4.01	2018-10-18	Remove: emv_terminal_param_set emv_terminal_param_set2 Add: emv_terminal_param_set_tlv	Michael Li
4.02	2018-10-22	Update emv_aid_param_add	Michael Li
4.03	2018-10-31	Add Appendix	Michael Li
4.04	2018-11-02	Update error code	Michael Li
4.05	2018-11-09	Add: 4.16 emv_get_kernel_checksum 4.17 emv_get_config_checksum	Michael Li

1. IC Reader

1.1 open reader and wait card

```
/*
* @param[in] reader : reader type        : 0   all of readers
*                                        : 1   only contact reader
*                                        : 2   only contactless reader
* return value          : < 0   Fail
*                         >= 0 Success
*     (If select open all of readers, any open success return success)
*/
int open_reader(int reader)
```

1.2 close reader

```
/*
* @param[in] reader:  reader type  : 0 all of readers
*                                  : 1 only contact reader
*                                  : 2 only contactless reader
*/
void close_reader(int reader)
```

1.3 get current card type

```
/*
* return value  : 1    contact card
*               : 2    contactless card
*               : -1   no card
*/
int get_card_type(void)
```

1.4 get card ATR

```
/*
* @param[out] pATR : the value of ATR
* return value  : the length of ATR
*/
int get_card_atr(unsigned char *pATR)
```

1.5 APDU command

```
/*
* @param[in] cmd       :APDU command
* @param[in] cmdLength : the length of APDU command
```

```
* @param[out] respData : the value of card response
* @param[in]   respDataLength : accepted max length of card reasponse
* return value  :  >= 0 :the length of card response
*                  < 0  :Fail
*/
int transmit_card(    unsigned char *cmd,
                      int cmdLength,
                      unsigned char *respData,
                      int respDataLength)
```

2. store and set EMV data

2.1 check the existence of data for the tag

```
/*
* @param[in] tag : tag name
* return value          : < 0     the data not exist
*                         >= 0    the length of data
*/
int emv_is_tag_present(int tag)
```

2.2 get the data for the tag

```
/*
* @param[in]  tag             : tag name
* @param[out] data            : the value of the data
* @param[in]  dataLength      : accepted max length of the data
* return value                : < 0 : Fail
*                               >= 0: the length of the data
*/
int emv_get_tag_data(int tag, unsigned char *data, int dataLength)
```

2.3 get the data for the tag list

```
/*
* @param[in]   tagNames              : the list of the tags
* @param[in]   tagCount              : the count of the tags
* @param[out] pTagsValue             : the values of the data（TLV format）
* @param[in]   pTagsValueLength      : accepted max length of the data
* return value                       : < 0 : Fail
*                                    : >= 0: the length of the data
*/
int emv_get_tag_list_data(int *tagNames, int tagCount,
                          unsigned char *pTagsValue,
                          int pTagsValueLength);
```

2.4 set the data for the tag

```
/*
* @param[in] tag    : tag name
* @param[in] data   : the value of the data
* @param[in] length : the length of the data
* return value     : < 0 : Fail
*                  : >= 0 : the tag的长度
*/
int emv_set_tag_data(int tag, unsigned char *data, int length)
```

3. EMV transaction processing

3.1 EMVKernel initialize

```
typedef struct
{
    // callback function for card event
    CARD_EVENT_OCCURED pCafdEventOccured;
    // callback function for EVM processing
    EMV_PROCESS_CALLBACK pEVMProcessCallback;
}EMV_INIT_DATA;
void emv_kernel_initialize(unsigned char *pInitData)

1）typedef void (*CARD_EVENT_OCCURED) (int eventType)
    // any card event occured, this function will be revoked
    // @param[in] eventType  : SMART_CARD_EVENT_INSERT_CARD = 0;
    //                       : SMART_CARD_EVENT_REMOVE_CARD = 1;
    //                       : SMART_CARD_EVENT_POWERON_ERROR = 9;
    //                       :SMART_CARD_EVENT_CONTALESS_HAVE_MORE_CARD = 10;
2）typedef void (*EMV_PROCESS_CALLBACK)(unsigned char *pData);
    // callback function for EVM processing，pData have 2 bytes
    // unsigned char status = pData[0]；
    // unsigned char desc = pData[1]；
* status：
*    STATUS_ERROR = 0; //ERROR
*    STATUS_CONTINUE = 1; // not completed, need to continue
*    STATUS_COMPLETION = 2; // completed
* desc
*    when status = STATUS_COMPLETION，desc means：
*        APPROVE_OFFLINE = 1;   //Transaction approved Offline
*        APPROVE_ONLINE = 2;    //Transaction approved Online
*        DECLINE_OFFLINE = 3; //Transaction declined Offline
*        DECLINE_ONLINE = 4; //Transaction declined Online
```

```
*
*     when status = STATUS_ERROR，desc means：
*          SUCCESS = 0; //SUCCESS
*          ERROR_NO_APP    =    1; //No Supported Application Selected
*          ERROR_CARD_BLOCKED = 2; //card return 6A81 when Application Select
*          ERROR_APP_SELECT = 3; //Error when Application Select
*          ERROR_INIT_APP = 4; //Error when Initialize Application Data
*          ERROR_EXPIRED_CARD =    5;    // Card Expired
*          ERROR_APP_DATA = 6; //Error when Read Application Data
*          ERROR_DATA_INVALID = 7; // have invalid data
*          ERROR_DATA_AUTH = 8; // Fail in offline authentication
*          ERROR_GEN_AC = 9; //Generate AC error when Transaction Process
*          ERROR_PROCESS_CMD = 10; //Process Command ERROR
*          ERROR_SERVICE_NOT_ALLOWED = 11; //Service not Allowed
*          ERROR_PINENTERY_TIMEOUT = 12; //PIN Entry timeout
*          ERROR_OFFLINE_VERIFY = 13; //Check Offline PIN Error when Cardholder Verify
*          ERROR_NEED_ADVICE = 14; //Communication Error with Host, but the card need
advice, halted the transaction
*          ERROR_USER_CANCELLED = 15;
*          ERROR_AMOUNT_OVER_LIMIT = 16; // amount over limit
*          ERROR_AMOUNT_ZERO = 17; // amount can not be zero
*          ERROR_OTHER_CARD = 18；    // Please try other card
*          ERROR_MISSING_DATA = 19；    //missing mandatory data
*          ERROR_APP_BLOCKED = 20；    // application is blocked
*          ERROR_POWER_ON_AGAIN = 21；    // Please power on card again
*          ERROR_CONTACTLESS_INTERRUPT = 22；    // contact card inserted when reading
contactless card record
*          ERROR_MSD_NOT_SUPPORTED = 30；    // Magstripe Mode not suported
*          ERROR_AMOUNT_NOT_PRESENT = 31；    // amount not present
*          ERROR_CCC = 32；    // CCC Error for mastercard contactless
*          ERROR_EXCHANGE_RR_DATA = 33；    // Exchange relay resistance data error for
mastercard contactless
*          ERROR_GET_PDOL_DATA = 34；    // Get PDOL data error
*          ERROR_RESTART = 35；    // Please restart the transaction
*          ERROR_SEE_PHONE = 36；    // Please see phone
*          ERROR_NEXT_AID = 37；    // Please select next aid
*          ERROR_ANOTHER_INTERFACE = 38；    // Please try another interface
*
*
*     when status = STATUS_CONTINUE，desc means：
*          EMV_CANDIDATE_LIST = 1; //notify Application show Application Candidate List
*          EMV_APP_SELECTED = 2; //Application Select Completed
*          EMV_READ_APP_DATA = 3; //Read Application Data Completed
*          EMV_DATA_AUTH = 4; //Data Authentication Completed
```

```
*          EMV_OFFLINE_PIN = 5; // notify Application prompt Caldholder enter offline PIN,
*          EMV_ONLINE_ENC_PIN = 6; //notify Application prompt Caldholder enter Online PIN
*          EMV_PIN_BYPASS_CONFIRM = 7; //notify Application confirm to Accepted PIN Bypass or not
*          EMV_PROCESS_ONLINE = 8; //notify Application to Process Online
*          EMV_ID_CHECK =    9; //notify Application Check Cardholder's Identification
*/
```

3.2 Initialize EMV transaction data

```
void emv_trans_initialize(void)
```

3.3 EMV processing function

```
/*
* return value:  >=0 SUCCESS, <0 Fail
*/
int emv_process_next(void)
```

4. Others functions

4.1 Get EMV Kernel version

```
/**
* @param[out] buffer:    the value of emv kernel version
* @param[in] bufferLength: accepted max length of emv kernel version
* return value:  the length of emv kernel verion
*/
int emv_get_version_string(unsigned char *buffer, int bufferLength)
```

4.2 Set transaction amount

```
/**
* @param[in] amount:    '\0' as ending mark
* return value:    >=0 Success; < 0 Fail
*/
int emv_set_trans_amount(unsigned char *amount)
```

4.3 Set other amount

```
/**
* @param[in] amount:    '\0' as ending mark
* return value:    >=0 Success; < 0 Fail
*/
int emv_set_other_amount(unsigned char *amount)
```

4.4 Set transaction type

```
int emv_set_trans_type(unsigned char transType)
```

#define TRANS_GOODS_SERVICE	0x00
#define TRANS_CASH	0x01
#define TRANS_INQUIRY	0x04
#define TRANS_TRANSFER	0x05
#define TRANS_PAYMENT	0x06
#define TRANS_ADMIN	0x07
#define TRANS_CASHBACK	0x09
#define TRANS_CARD_RECORD	0x0A

4.5 set emv kernel type

```
/**
* @param[in] kernelType:  1    EMV Contact Kernel
*                         2    EMV Contactless Kernel
*                         3    UPCASH Kernel for China Union Pay
*/
int emv_set_kernel_type(unsigned char kernelType)
```

4.6 Is needed advice the transaction

```
/**
* return value:  1 need advice
*                0 not need advice
*/
int emv_is_need_advice(void)
```

4.7 Is needed sign the transaction

```
/**
* return value:  1 need sign
*                0 not need sign
*/
int emv_is_need_signature(void)
```

4.8 Set the parameter for force online

```
/**
* @param[in] flag:  flag=1 Yes,  flag = 0 No
*/
int emv_set_force_online(int flag)
```

4.9 Read transaction record from the card

```
/**
* @param[out] data          : transaction record
* @param[in] dataLength     : accepted max length for the transaction record
* return value              : < 0 : Fail
*                           : >= 0: record count
*/
int emv_get_card_record(uint8_t *data, int dataLength)
```

4.10 Get application list

```
/*
* @param[out] data : application list as "LV" format
* @param[in] dataLength :  accepted max length for application list
* return value          : < 0 : Fail
*                       : >= 0: application count
*/
int emv_get_candidate_list(uint8_t *data, int dataLength)
```

4.11 Set the selected index for application selection

```
/**
* @param[in] index :  the selected index (started by 0)
* return value  : < 0 : Fail
*               : >= 0: Success
*/
int emv_set_candidate_list_result(int index)
```

4.12 Set the result for cardholder ID check

```
/* ID Type（9F62）、ID Number(9F61)
* @param[in] result :  0: check Fail, 1:check success
* return value  : < 0 : Fail
*               : >= 0: Success
*/
int emv_set_id_check_result(int result)
```

4.13 Set the result for Online PIN

```
/**
* @param[in] result : 0: Online PIN not input, 1:Online PIN inputted
* return value      : < 0 : Fail
*                   : >= 0: Success
*/
int emv_set_online_pin_entered(int result)
```

4.14 Set acceptance for Bypass PIN

```
/**
```

```
* @param[in] result :  0: refused bypass pin
                       1: accepted bypass pin
* return value      : < 0 : Fail
*                   : >= 0: Success
*/
int emv_set_pin_bypass_confirmed(int result)
```

4.15 Set the result for online certification

```
/**
* @param[in] result :   -1:communication failed；0: host refused；1: host accepted
* @param[in] respCode : 2 bytes response code from the host
* @param[in] issuerRespData : the emv data from the host
* @param[in] issuerRespDataLength : the length of the emv data from the host
* return value      : < 0 : Fail
*                   : >= 0: Success
*/
int emv_set_online_result(int result,
                          unsigned char *respCode,
                          unsigned char *issuerRespData,
                          int issuerRespDataLength)
```

4.16 Get Kernel checksum

```
/**
* @param[out] buffer:    the value of emv kernel checksum
* @param[in] bufferLength：accepted max length
* return value:  the length of kernel checksum
*/
int emv_get_kernel_checksum(unsigned char *buffer, int bufferLength)
```

4.17 Get Configuration checksum

```
/**
* @param[out] buffer:    the value of configuration checksum
* @param[in] bufferLength：accepted max length
* return value:  the length of configuration checksum
*/
int emv_get_config_checksum(unsigned char *buffer, int bufferLength)
```

5. Setup EMVparameters

5.1 Clear AID info

```
/**
```

```
* return value:  >=0: Success; < 0: Fail
*/
int emv_aidparam_clear(void)
```

5.2 Add AID info

```
/*
* @param[in] data : see form below, format is TLV
* @param[in] dataLength : the length of the data
* return value       : < 0 : Fail
*                    : >= 0: Success
*/
int emv_aidparam_add( uint8_t *data, int dataLength)
```

name	Format	length（byte）	tag
AID	b	5－16	9F06
Application selection Indicator（ASI）	b	1	DF01
Application version number	b	2	9F08
TAC－Default	b	5	DF11
TAC－Online	b	5	DF12
TAC－Denial	b	5	DF13
Terminal floor limit	b	4	9F1B
Threshold value for Biased Random Selection	b	4	DF15
Maximum Target Percentage to be used for Biased Random Selection	cn	1	DF16
Target Percentage to be used for Random Selection	cn	1	DF17
Default DDOL	b	Var.	DF14
Ability for Online PIN	b	1	DF18
Application Label	an	1-16	50
Application Preferred Name	an	1-16	9F12
Application Priority Indicator	b	1	87
Merchant Identifier	an	15	9F16
Acquirer Identifier	n	6-11	9F01
MCC	n	4	9F15
POS Entry Mode	n	2	9F39

name	Format	length（byte）	tag
Transaction Reference Currency Code	n	3	9F3C
Transaction Reference Currency Exponent	n	1	9F3D
Default TDOL	b	Var.	DF22
Contactless Floor Limit	n	6	DF19
Contactless Limit	n	6	DF20
CVM Limit	n	6	DF21
Kernel ID	n	1	DF810C
C2: CVM Capability – CVM Required	b	1	DF8118
C2: CVM Capability – No CVM Required	b	1	DF8119
C2: kernel configuration	b	2	DF811B
C2: Mag-stripe CVM Capability – CVM Required	b	1	DF811E
C2: Reader Contactless transaction limit (No On-device CVM)	n	6	DF8124
C2: Reader Contactless transaction limit (On-device CVM)	n	6	DF8125
C2: Mag-stripe CVM Capability – No CVM Required	b	1	DF812C

* C2 – Only for Mastercard Payapss

5.3 Clear CAPK info

```
/**
* return value:  >=0 Success; < 0 Fail
*/
int emv_capkparam_clear(void)
```

5.4 Add CAPK info

```
/*
* @param[in] data : see form below, format is TLV
* @param[in] dataLength : the length of the data
* return value      : < 0 : Fail
*                   : >= 0: Success
```

```
*/
int emv_capkparam_add( uint8_t *data, int dataLength)
```

Name	Format	length（byte）	tag
RID	b	5	9F06
Certification Authority Public Key Index	b	1	9F22
Certification Authority Public Key Expiration Date	n8	8	DF05
Certification Authority Public Key hash Algorithm Indicator	b	1	DF06
Certification Authority Public Key Algorithm Indicator	b	1	DF07
Certification Authority Public Key Modulus	b	Var.	DF02
Certification Authority Public Key Exponent	b	1 or 3	DF04
Certification Authority Public Key Checksum	b	Var.	DF03

5.5 Set EMV terminal parameters by TLV

Supported Tag	Description
5F2A	Transaction Currency Code
5F36	Transaction Currency Exponent
9F16	Merchant Identification
9F1A	Terminal Country Code
9F1C	Terminal Identification
9F1E	IFD Serial Number
9F33	Terminal Capabilities
9F35	Terminal Type
9F40	Additional Terminal Capabilities
9F4E	Merchant Name and Location
9F66	TTQ first byte
DF19	Contactless floor limit
DF20	Contactless transaction limit
DF21	CVM limit
DF8104	Balance Read Before Gen AC (C2)
DF8105	Balance Read After Gen AC (C2)
DF811C	Max Lifetime of Torn Transaction Log Record (C2)
DF811D	Max Number of Torn Transaction Log Records (C2)
DF812D	Message Hold Time (C2)
DF8132	Minimum Relay Resistance Grace Period (C2)
DF8133	Maximum Relay Resistance Grace Period (C2)
DF8134	Terminal Expected Transmission Time For Relay Resistance C-APDU (C2)

DF8135	Terminal Expected Transmission Time For Relay Resistance R-APDU (C2)
DF8136	Relay Resistance Accuracy Threshold (C2)
DF8137	Relay Resistance Transmission Time Mismatch Threshold (C2)
EF01	Status check support: 1 – Support; 0 - No
EF02	Zero check support: 1 – Support; 0 - No
EF03	Authorization Type For American Expresspay(C4): 0-Immediate; 1-Delayed
EF04	CDCVM support: 1 – Support; 0 - No
EF05	Extended Selection: 1 – Support; 0 - No

```
int emv_terminal_param_set_tlv( uint8_t *data, int dataLength)
```

5.6 Clear Exception File

```
/**
* return value:  >=0 Success; < 0 Fail
*/
int emv_exception_file_clear(void)
```

5.8 Add Exception File

```
Typedef struct{
    unsigned char cardNo[19];        // PAN
    unsigned char panSequence;     // PAN Sequence Number
}ExceptionFile
int emv_exception_file_add( unsigned char *exceptFile)
```

5.9 Clear Revoked Certicates

```
/**
* return value:  >=0 Success; < 0 Fail
*/

int emv_revoked_cert_clear(void)
```

5.10 Add revoked Certificate

```
Typedef struct{
    unsigned char rid[5];
    unsigned char capki;
}RevokedCert
int emv_revoked_cert_add( uint8_t *revokedCert)
```

6. Addendix A

A. 1 CVM Capability – CVM Required

Tag: 'DF8118'
Length: 1
Format: b
Description: Indicates the CVM capability of the Terminal and Reader when the transaction amount is greater than the *Reader CVM Required Limit*.

CVM Capability – CVM Required		
Byte 1	b8	Plaintext PIN for ICC verification
	b7	Enciphered PIN for online verification
	b6	Signature (paper)
	b5	Enciphered PIN for offline verification
	b4	No CVM required
	b3-1	Each bit RFU

A. 2 CVM Capability – No CVM Required

Tag: 'DF8119'
Length: 1
Format: b
Description: Indicates the CVM capability of the Terminal and Reader when the transaction amount is less than or equal to the *Reader CVM Required Limit*.

CVM Capability – No CVM Required		
Byte 1	b8	Plaintext PIN for ICC verification
	b7	Enciphered PIN for online verification
	b6	Signature (paper)
	b5	Enciphered PIN for offline verification
	b4	No CVM required
	b3-1	Each bit RFU

A. 3 Kernel Configuration

Tag: 'DF811B'
Length: 1
Format: b
Description: Indicates the Kernel configuration options.

Kernel Configuration		
Byte 1	b8	Mag-stripe mode contactless transactions not supported
	b7	EMV mode contactless transactions not supported
	b6	On device cardholder verification supported
	b5	Relay resistance protocol supported
	b4-1	Each bit RFU

A. 4 Mag-stripe CVM Capability – CVM Required

Tag: 'DF811E'
Length: 1
Format: b
Description: Indicates the CVM capability of the Terminal/Reader in the case of a mag-stripe mode transaction when the *Amount, Authorized (Numeric)* is greater than the *Reader CVM Required Limit*.

Mag-stripe CVM Capability – CVM Required			
Byte 1	b8-5	CVM	
			0000: NO CVM
			0001: OBTAIN SIGNATURE
			0010: ONLINE PIN
			1111: N/A
			Other values: RFU
	b4-1	Each bit RFU	

A. 5 Mag-stripe CVM Capability – No CVM Required

Tag: 'DF812C
Length: 1
Format: b
Description: Indicates the CVM capability of the Terminal/Reader in the case of a mag-stripe mode transaction when the *Amount, Authorized (Numeric)* is less than or equal to the *Reader CVM Required Limit*.

Mag-stripe CVM Capability – No CVM Required			
Byte 1	b8-5	CVM	
			0000: NO CVM
			0001: OBTAIN SIGNATURE
			0010: ONLINE PIN
			1111: N/A
			Other values: RFU
	b4-1	Each bit RFU	