NEWPOS EMV Kernel Programmer User Manual

V2.2

Author: Harrison Lee

Date: 2015-04-20

1. Document History

Date	Version	Author	Comments
2011-9-8	V1.0.0	Harrison Lee	First Draft
2013-12-14	V2.0.0	Harrison Lee	Add the VISA contactless API
2015-03-02	V2.1.0	Harrison Lee	Add callback function for the CAPK and Terminal AID
2015-04-20	V2.2.0	Harrison Lee	Add EMV Contactless and Paypass API

2. Contents

Common Error Code Definition

```
enum emv_error_code{
    EMV_ERRNO_INVAL = 2048,           //Invalid value
    EMV_ERRNO_NOMEM,                  //Memory not enough
    EMV_ERRNO_NODATA,                 //Data missing
    EMV_ERRNO_DATE,                   //Date format incorrect
    EMV_ERRNO_CHECKSUM,               //CAPK checksum incorrect
    EMV_ERRNO_EXPRIED = 2048+5,          //Application expired
    EMV_ERRNO_TLV,                    //TLV format incorrect
    EMV_ERRNO_UNKNOWN_TAG,            //Unknow tag
    EMV_ERRNO_EXISTED,                //Tag duplicated or Tag exists already
    EMV_ERRNO_LEN,                    //Data length incorrect
    EMV_ERRNO_PIN = 2048+10,             //PIN data incorrect
    EMV_ERRNO_KEY,                    //RSA calculate failed
    EMV_ERRNO_SW,                  //Card response incorrect status word
    EMV_ERRNO_DATA,                   //Data format incorrrect
    EMV_ERRNO_CARD_BLOCKED,           //User card blocked
    EMV_ERRNO_APP_BLOCKED = 2048+15,  //Application blocked
    EMV_ERRNO_NO_APP,                 //There no application on the card
    EMV_ERRNO_CANCEL,                 //Transaction canceled by operator
    EMV_ERRNO_NO_ACCEPTED,            //Transaction APPROVED
    EMV_ERRNO_PIN_BLOCKED,           //Offline PIN blocked
    EMV_ERRNO_BYPASS_PIN = 2048+20,     //PIN required and user no entered.
    EMV_ERRNO_DECLINE,                //Transaction was DECLINED
    EMV_ERRNO_FORCE_ACCEPTED,        //FORCE ACCEPETED
    EMV_ERRNO_ONLINE_ERROR,       //Could not connect to the issuer host
    EMV_ERRNO_ONLINE_TIMEOUT,         //online auth. timeout
    EMV_ERRNO_SERVICE_NOT_ALLOWED,  //Service not allowed
};
```

3. Terminal Application Management APIs

3.1. AID Element Structure

```
struct terminal_aid_info
{
    uint8_t aid_len;       //Length of AID
    uint8_t aid[16];     //AID
    uint8_t support_partial_aid_select;//Support partial AID selection
```

```
    uint8_t application_priority;           //Application priority
    uint32_t target_percentage;           //Target percentage
    uint32_t maximum_target_percentage;     //Maximum target percantage
    uint32_t terminal_floor_limit;          //Terminal floor limit
    uint32_t threshold_value;               //Threshold value
    uint8_t terminal_action_code_denial[5];  //terminal action code – denial
    uint8_t terminal_action_code_online[5];  //terminal action code – online
    uint8_t terminal_action_code_default[5]; //terminal action code – default
    uint8_t acquirer_identifier[6];         //acquirer identifier
    uint8_t len_of_default_ddol;               //length of default DDOL
    uint8_t default_ddol[254];               //Default DDOL
    uint8_t len_of_default_tdol;              //Length of default TDOL
    uint8_t default_tdol[254];               //Default TDOL
    uint8_t application_version[2];          //Application version number
    uint8_t len_of_terminal_risk_management_data; //Length of terminal risk
management data
    uint8_t  terminal_risk_management_data[8]; //Terminal risk management
data
    uint32_t cl_ReaderMaxTransAmount;    //Contact    less    Reader    max
transaction amount
    uint32_t cl_Floor_Limit;        //Contactless reader offline floor limit
    uint32_t cl_CVM_Amount;           //Contact less CVM amount
};
```

3.2. fn_callback_emv_terminal_aid_count

Prototype	typedef int (*fn_callback_emv_terminal_aid_count)(void);	
Description	Retrieve the terminal AID count	
Parameters	NONE	
Return Value	0: success other: Failed	

Example	int callback_emv_terminal_aid_count(void)
	{
	return 10;
	}	

3.3. fn_callback_emv_terminal_aid_get

Prototype	typedef int (*fn_callback_emv_terminal_aid_get)(int index, void * pAidInfo, unsigned int *puiSize);	
Description	Retrieve the terminal AID by the index	
Parameters	index	Terminal AID Storage index

	pAidInfo	Buffer used to save the AID information
	puiSize	Pass in the buffer size, pass out the size of the AID information.
Return Value	0: success other: Failed	

Example	int callback_emv_terminal_aid_get(int index, void * pAidInfo, unsigned int *puiSize);
	{
	int retval = 0;
	memcpy(pAidInfo, Aidinfo, sizeof(*AidInfo));
	*puiSize = sizeof(*AidInfo);
	return retval;
	}	

3.4. emv_aid_manage_set_callback

Prototype	**void emv_aid_manage_set_callback(fn_callback_emv_terminal_aid_count callback_emv_terminal_aid_count, fn_callback_emv_terminal_aid_get callback_emv_terminal_aid_get);**	
Description	Set AID callback function.	
Parameters	callback_emv_terminal_aid_count	Callback function pointer, this function will return number of the aid count.
	callback_emv_terminal_aid_get	Callback function pointer, this function will used to retrieve a specified index of the terminal AID.
Return Value	NONE	
Example	emv_aid_manage_set_callback(callback_emv_terminal_aid_count, callback_emv_terminal_aid_get);	

3.5. emv_delete_all_terminal_aids

Prototype	**void emv_delete_all_terminal_aids(void)**	
Description	Delete All AIDs from terminal	
Parameters	NONE	
Return	NONE	

Value	
Example	emv_delete_all_terminal_aids();

3.6. emv_add_one_aid_info

Prototype	**int emv_add_one_aid_info(const struct terminal_aid_info *info)**
Description	Add one AID to kernel
Parameters	info: AID infomation
Return Value	0: success Other:failed

Example	struct terminal_aid_info aidinfo =
	{
	7,
	{0xA0,0x00,0x00,0x00,0x03,0x10,0x10},
	0x01,
	0x00,
	0,
	0,
	10000,
	100,
	{0x00,0x00,0x00,0x00,0x00},
	{0x00,0x00,0x00,0x00,0x00},
	{0x00,0x00,0x00,0x00,0x00},
	"NEWPOS",
	3,
	{0x9F,0x37,0x04},
	3,
	{0x9F,0x37,0x04},
	{0x00,0x8C},
	0x00,
	{0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00},
	20000,
	10000,
	5000,
	};
	
	if (0 == emv_add_one_aid_info(&aidinfo))
	{
	//Success
	}
	Else
	{

	//Add failed }

3.7. emv_get_one_aid_info

Prototype	**int emv_get_one_aid_info(int index, struct terminal_aid_info *info);**	
Description	Get AID information with the index.	
Parameters	index	Index of the aid
	info	Aid info
Return Value	0:Success Other:Failed	

Example	struct terminal_aid_info aidinfo;
	int i, count;
	for (i=0; i < count; i++)
	{
	if (0 == emv_get_one_aid_info(i, &aidinfo))
	{
	//Success
	}
	else
	{
	//Add failed
	}
	}	

3.8. emv_del_one_aid_info

Prototype	**int emv_del_one_aid_info(uint8_t aid_len, const void *aid);**	
Description	Get AID information with the index.	
Parameters	aid_len	Length of the AID
	aid	Point to the AID buffer
Return Value	0:Success Other:Failed	

Example	const uint8_t * paid = "\xA0\x00\x00\x00\x03\x01\x01"
	uint8_t iaid = 7;
	if (0 == emv_del_one_aid_info(iaid, paid))
	{
	//Success	

	}
	else
	{
	//Failed
	}

3.9. emv_get_aid_info_num

Prototype	**int emv_get_aid_info_num(void);**
Description	Get number of aids
Parameters	NONE
Return Value	Number of aids
Example	int iaidcount = 0; iaidcount = emv_get_aid_info_num();

3.10. emv_check_aid_in_terminal

Prototype	**int emv_check_aid_in_terminal(uint8_t aid_len, const void *aid);**	
Description	Chek is the aid present in the kernel.	
Parameters	aidlen	Length of the AID
	aid	Point to the AID buffer.
Return Value	0: Present in the kernel Other:Not present	

Example	const uint8_t * paid = "\xA0\x00\x00\x00\x03\x01\x01"
	uint8_t iaid = 7;
	if (0 == emv_check_aid_in_terminal(iaid, paid))
	{
	./Present
	}
	else
	{
	//Not present
	}	

4. CAPK Management APIs

4.1. CAPK Element Structure

```
struct issuer_ca_public_key
{
    uint8_t RID[5];              //RID
    uint8_t Index;             //CAPK Index
    uint8_t len_of_modulus;      //Modules Length
    uint8_t modulus[248];        //Modules
    uint8_t len_of_exponent;      //Exponents length
    uint8_t exponent[3];          //Exponents
    uint8_t expiration_date[3];     //Expire Date
    uint8_t checksum[20];    //Checksum HASH(RID + CAPK Index + Modules +
Exponents
};
```

4.2. fn_callback_emv_capk_count

Prototype	typedef int (*fn_callback_emv_capk_count)(void);	
Description	Callback function to get the CAPK count	
Parameters	NONE	
Return Value	>=0: success Oter: failed.	

Example	int callback_emv_capk_count(void)
	{
	int retval = 0;
	retval = 10;
	return retval;
	}	

4.3. fn_callback_emv_capk_get

Prototype	typedef int (*fn_callback_emv_capk_get)(int index, struct issuer_ca_public_key *key);	
Description	Callback function to retrieve a CAPK by index	
Parameters	Index	Storage index of the CAPK
	Key	Key buffer to save the CAPK
Return Value	0: success Oter: failed.	
Example	int callback_emv_capk_get(int index, struct	

	issuer_ca_public_key * key)
	{
	int retval = 0;
	struct issuer_ca_public_key * key_val[10];
	memcpy(key, key_val[index], sizeof(*key));
	Return retval;
	}

4.4. emv_capk_set_callback

Prototype	**void emv_capk_set_callback(fn_callback_emv_capk_count callback_emv_capk_count, fn_callback_emv_capk_get callback_emv_capk_get)**	
Description	Set the CAPK callback functions.	
Parameters	callback_emv_capk_count	Return the CAPK count
	callback_emv_capk_get	To retrieve a CAPK by index
Return Value	NONE	
Example	emv_capk_set_callback(emv_capk_get_count, emv_capk_get);	

4.5. emv_delete_all_ca_public_keys

Prototype	**void emv_delete_all_ca_public_keys(void);**	
Description	Delete All CAPK from terminal	
Parameters	NONE	
Return Value	NONE	
Example	emv_delete_all_ca_public_keys ();	

4.6. emv_add_one_ca_public_key

Prototype	**int emv_add_one_ca_public_key(const struct issuer_ca_public_key *key);**	
Description	Add on CAPK to kernel	
Parameters	key	CA Public key
Return Value	0:success Other:Failed	
Example	struct issuer_ca_public_key key = { {0xA0,0x00,0x00,0x00,0x03}, 0x50,	

```
    128,
{0xD1,0x11,0x97,0x59,0x00,0x57,0xB8,0x41,0x96,0xC2
,0xF4,0xD1,0x1A,0x8F,0x3C,0x05,
0x40,0x8F,0x42,0x2A,0x35,0xD7,0x02,0xF9,0x01,0x06,
0xEA,0x5B,0x01,0x9B,0xB2,0x8A,
0xE6,0x07,0xAA,0x9C,0xDE,0xBC,0xD0,0xD8,0x1A,0x38,
0xD4,0x8C,0x7E,0xBB,0x00,0x62,
0xD2,0x87,0x36,0x9E,0xC0,0xC4,0x21,0x24,0x24,0x6A,
0xC3,0x0D,0x80,0xCD,0x60,0x2A,
0xB7,0x23,0x8D,0x51,0x08,0x4D,0xED,0x46,0x98,0x16,
0x2C,0x59,0xD2,0x5E,0xAC,0x1E,
0x66,0x25,0x5B,0x4D,0xB2,0x35,0x25,0x26,0xEF,0x09,
0x82,0xC3,0xB8,0xAD,0x3D,0x1C,
0xCE,0x85,0xB0,0x1D,0xB5,0x78,0x8E,0x75,0xE0,0x9F,
0x44,0xBE,0x73,0x61,0x36,0x6D,
0xEF,0x9D,0x1E,0x13,0x17,0xB0,0x5E,0x5D,0x0F,0xF5,
0x29,0x0F,0x88,0xA0,0xDB,0x47},
    3,
   {0x01,0x00,0x01},
   {0x15,0x12,0x31},
{0xB7,0x69,0x77,0x56,0x68,0xCA,0xCB,0x5D,0x22,0xA6
,0x47,0xD1,0xD9,0x93,0x14,0x1E,
    0xDA,0xB7,0x23,0x7B}
};
if (0 == emv_add_one_ca_public_key(&key))
{
   //Success
}
else
{
   //Failed
}
```

4.7. emv_get_one_ca_public_key

Prototype	**int emv_get_one_ca_public_key(const void *RID, uint8_t index, struct issuer_ca_public_key *key);**	
Description	Search and return the CA Public key	
Parameters	RID	RID
	index	CA Public key index
	key	Return key value
Return	0: Success Other:Failed	

Value	

Example	const uint8_t * pRID = "\xA0\x00\x00\x00\x01";
	uint8_t index = 0x50;
	struct issuer_ca_public_key key;
	if (0 == emv_get_one_ca_public_key(pRID, index, &key))
	{
	//Success
	}
	else
	{
	//Failed
	}

4.8. emv_get_one_ca_public_key_by_no

Prototype	**int emv_get_one_ca_public_key_by_no(int no, struct issuer_ca_public_key *key);**	
Description	Get a CA Public key with the index.	
Parameters	no	Index
	key	Return key value
Return Value	0: Success Other:Failed	

Example	int i, count;
	struct issuer_ca_public_key key;
	count = emv_get_ca_public_key_num();
	for (i = 0; i < count; i++)
	{
	if (0 == emv_get_one_ca_public_key_by_no (i, &key))
	{
	//Success
	}
	else
	{
	//Failed
	}
	}	

4.9. emv_delete_one_ca_public_key

Prototype	**int emv_delete_one_ca_public_key(const void *RID,**

	uint8_t index);	
Description	Delete a CA public key	
Parameters	RID	RID
	index	CA Public key index
Return Value	0: Success Other:Failed	

Example	const uint8_t * pRID = "\xA0\x00\x00\x00\x01";
	uint8_t index = 0x50;
	if (0 == emv_delete_one_ca_public_key (pRID, index))
	{
	//Success
	}
	else
	{
	//Failed
	}	

4.10. emv_get_ca_public_key_num

Prototype	**int emv_get_ca_public_key_num(void);**
Description	Get number of CAPKs
Parameters	NONE
Return Value	Number of CA Public keys
Example	int count; count = emv_get_ca_public_key_num();

5. Kernel Configuration

```
struct terminal_mck_configure
{
    uint8_t terminal_type;
    uint8_t terminal_capabilities[3];
    uint8_t additional_terminal_capabilities[5];

    uint8_t support_pse_selection_method;
    uint8_t support_cardholder_confirmation;
    uint8_t support_certificate_revocation;
```

```
    uint8_t support_default_ddol;

    uint8_t support_bypass_pin_entry;
    uint8_t support_get_pin_try_counter;

    uint8_t support_floor_limit_checking;
    uint8_t support_random_transaction_selection;
    uint8_t support_velocity_checking;
    uint8_t support_transaction_log;
    uint8_t support_exception_file;

    uint8_t support_terminal_action_codes;
    uint8_t support_forced_online_capability;
    uint8_t support_advices;
    uint8_t support_issuer_initiated_voice_referrals;
    uint8_t support_batch_data_capture;
    uint8_t support_online_data_capture;
    uint8_t support_default_tdol;

    uint8_t support_multi_language;
    uint8_t support_forced_acceptance_capability;
    uint8_t support_card_initiated_voice_referrals;
    uint8_t support_trm_based_on_aip;

    uint8_t pos_entry_mode;
    uint8_t checksum[20];
}; // 1-> Support 0 -> No Support
```

5.1. emv_set_mck_configure

Prototype	**int emv_set_mck_configure(struct terminal_mck_configure *config);**	
Description	Set kernel parameters.	
Parameters	config	Kernel parameters
Return Value	0: Success Other:Failed	

Example	struct terminal_mck_configure config;
	//Set the parameters.
	if (0 == emv_set_mck_configure(&config))
	{
	  //Success
	}
	Else	

	{
	//Failed
	}

5.2. emv_get_mck_configure

Prototype	**int emv_get_mck_configure(struct terminal_mck_configure *config);**	
Description	Get kernel parameters.	
Parameters	config	Kernel parameters
Return Value	0: Success Other:Failed	

Example	struct terminal_mck_configure config;
	if (0 == emv_get_mck_configure (&config))
	{
	//Success
	}
	else
	{
	//Failed
	}	

6. Kernel Type Setup

```
#define KERNEL_TYPE_EMV        0
#define KERNEL_TYPE_PBOC       1
#define KERNEL_TYPE_VISA       2
#define KERNEL_TYPE_MASTERCARD  3
#define KERNEL_TYPE_JCB        4
#define KERNEL_TYPE_AMEX       5
```

6.1. emv_set_kernel_type

Prototype	**void emv_set_kernel_type(int mode);**	
Description	Set Kernel Type	
Parameters	mode	KERNEL_TYPE_EMV KERNEL_TYPE_PBOC KERNEL_TYPE_VISA KERNEL_TYPE_MASTERCARD

		KERNEL_TYPE_JCB KERNEL_TYPE_AMEX
Return Value	NONE	
Example	emv_set_kernel_type(KERNEL_TYPE_EMV);	

6.2. emv_get_kernel_type

Prototype	**int emv_get_kernel_type(void);**
Description	Set Kernel Type
Parameters	NONE
Return Value	KERNEL_TYPE_EMV KERNEL_TYPE_PBOC KERNEL_TYPE_VISA KERNEL_TYPE_MASTERCARD KERNEL_TYPE_JCB KERNEL_TYPE_AMEX
Example	int mode; mode = emv_get_kernel_type();

7. Data Management

7.1. emv_init_data_element

Prototype	**void emv_init_data_element(void);**
Description	Init memory buffer for kernel. This API must be called before transaction API call.
Parameters	NONE
Return Value	NONE
Example	emv_init_data_element();

7.2. emv_get_data_element

Prototype	**int emv_get_data_element(const void *tag, uint8_t**

	taglen, uint8_t *len, void *val);	
Description	Get data value from kernel.	
Parameters	tag	Tag number buffer pointer
	taglen	Length of Tag
	len	Return value length
	val	Return value
Return Value	0: success 1:Failed	

Example	const uint8_t * pTag = "\x82";
	uint8_t iTagLen= 1;
	uint8_t ValueLen;
	uint8_t Value[255];
	if (0 == emv_get_data_element(pTag, iTagLen, &ValueLen, Value))
	{
	//Success
	}
	Else
	{
	//Failed
	}	

7.3. emv_set_data_element

Prototype	**int emv_set_data_element(const void *tag, uint8_t taglen, uint8_t len, const void *val);**	
Description	Set data value to kernel.	
Parameters	tag	Tag number buffer pointer
	taglen	Length of Tag
	len	Length of value
	val	Value
Return Value	0: success 1:Failed	

Example	const uint8_t * pTag = "\x9F\x02";
	uint8_t iTagLen= 2;
	uint8_t ValueLen = 6;
	uint8_t * pValue = "\x00\x00\x00\x10\x00\x00";
	if (0 == emv_set_data_element(pTag, iTagLen, ValueLen, Value))	

	{
	  //Success
	}
	Else
	{
	  //Failed
	}

7.4. emv_del_data_element

Prototype	**int emv_del_data_element(const void *tag, uint8_t taglen);**	
Description	Delete data element from kernel.	
Parameters	tag	Tag number buffer pointer
	taglen	Length of Tag
Return Value	0: success 1:Failed	

Example	const uint8_t * pTag = "\x9F\x02";
	uint8_t iTagLen= 2;
	if (0 == emv_del_data_element (pTag, iTagLen))
	{
	  //Success
	}
	Else
	{
	  //Failed
	}	

7.5. emv_check_data_element

Prototype	**int emv_check_data_element(const void *tag, uint8_t taglen);**	
Description	Check is the data element present or not	
Parameters	tag	Tag number buffer pointer
	taglen	Length of Tag
Return Value	0: Present Other: not present	
Example	const uint8_t * pTag = "\x9F\x02"; uint8_t iTagLen= 2;	

	if (0 == emv_check_data_element (pTag, iTagLen))
	{
	//Present
	}
	Else
	{
	//Not Present
	}

8. Callback Functions

8.1. Callback function structure

```
struct emv_callback_func{
    int (*emv_candidate_apps_selection)(void);
    void(*emv_multi_language_selection)(void);
    int (*emv_get_amount)(uint32_t *, uint32_t *);
    int (*emv_get_pin)(void *);
    int (*emv_return_pin_verify_result)(uint8_t);
    int (*emv_check_online_pin)(void);
    int (*emv_check_certificate)(void);
    int (*emv_online_transaction_process)(uint8_t *,uint8_t *, int *, uint8_t
*, int *,
                                    uint8_t *, int *,int *);
    int (*emv_issuer_referral_process)(void);
    int (*emv_advice_process)(int);

    int (*emv_check_revocation_certificate)(uint8_t, const void *, const void
*);
    int (*emv_check_exception_file)(uint8_t, const void *, uint8_t);
    int (*emv_get_transaction_log_amount)(uint8_t, const void *, uint8_t);
};
```

Prototype	**int emv_candidate_apps_selection (void)**
Description	provide all candidate application list for operator to select, and return the selected application to emv kernel.
Parameter	None

Return Value	0,1,2… : Success, return the selected app index. -1: Error Set errno EMV_ERRNO_NO_APP: no application EMV_ERRNO_CANCEL: Canceled by operator

Example	int emvSelCandApp(void)
	{
	call emv_get_all_candapps() to get all candidate applications supported both by terminal and ICC;
	prompt operator to select one application, and return the application index to emv kernel.
	if appcount is 0 then return -1 and set errno= EMV_ERRNO_NO_APP
	if canceled by operator return -1 and set errno= EMV_ERRNO_CANCEL
	}

8.2. emv_multi_language_selection

Prototype	**void emv_multi_language_selection(void)**
Description	EMV multi language process
Parameter	None
Return Value	None

Example:	void emv_language_selection (void)
	{
	1,Call emv_check_data_element() to get Tag 5F2D to view the languages supported by ICC;
	2,Check if there is exist a language supported both by ICC and terminal, if yes, then use this language for this emv transaction;
	3,otherwise, provide language list supported by terminal for operator to select, and use the language selected by operator for this emv transaction;
	}

8.3. emv_get_amount

Prototype	**int emv_get_amount(uint32_t * Amount, uint32_t * AmountOther)**

Description	Enter the transaction amount	
Parameter	Amount	Return the transaction Amount
	AmountOther	Options paramenter, if NULL then no need enter this value
Return Value	0: success -1: Failed Set errno to EMV_ERRNO_CANCEL	

Example	int emv_get_amount(uint32_t * Amount, uint32_t * AmountOther)
	{
	if enter amount success
	*Amount = amount_number
	else if canceled
	{
	errno = EMV_ERRNO_CANCEL;
	return -1
	}
	
	if (NULL != AmountOther)
	{
	if enter amount success
	*AmountOther = amount_number
	else if canceled
	{
	errno = EMV_ERRNO_CANCEL;
	return -1
	}
	}
	Return 0;
	}	

8.4. emv_get_pin

Prototype	**int emv_get_pin(void * pPIN);**	
Description	Enter the offline PIN	
Parameter	PIN	Return the clear PIN. Like "1234" "123456"……

Return Value	0: success -1: Failed If canceled Errno = EMV_ERRNO_CANCEL If BypassPIN Errno = EMV_ERRNO_BYPASS_PIN

Example	int emv_get_pin(void * pPIN)
	{
	//enter pin on PINPAD or KeyPad
	//save the pin value to pPIN
	if canceled
	{
	errno = EMV_ERRNO_CANCEL;
	return -1;
	}
	else if ByPass PIN
	{
	errno = EMV_ERRNO_BYPASS_PIN;
	return -1;
	}
	Strcpy(pPIN, "123456");
	return 0;
	}

8.5. emv_return_pin_verify_result

Prototype	**int emv_return_pin_verify_result(uint8_t iPinTryCounter);**	
Description	Display pin verify status	
Parameter	iPinTryCounter	

Return Value	0: success
	-1: Failed
	If canceled
	{
	Errno = EMV_ERRNO_CANCEL
	Return -1;
	}
	Else
	{
	Retrn 0;	

	}

Example	int retval=0;
	
	if(TryCounter==0){
	lcdCls();
	if(gl_SupCN == 0){
	lcdDisplay(0,3,DISP_CFONT|DISP_MEDIACY,"PIN OK");
	}else{
	lcdDisplay(0,3,DISP_CFONT|DISP_MEDIACY,"密码正确");
	}
	return 0;
	}
	if (TryCounter > 1)
	{
	return 0;
	}
	lcdCls();
	if(gl_SupCN == 0){
	lcdDisplay(0,1,DISP_CFONT|DISP_MEDIACY,"PIN ERROR");
	lcdDisplay(0,3,DISP_CFONT|DISP_MEDIACY,"Remain %d Times",TryCounter);
	lcdDisplay(0,5,DISP_CFONT|DISP_MEDIACY,"TRY AGAIN?");
	}else{
	lcdDisplay(0,1,DISP_CFONT|DISP_MEDIACY,"密码错误");
	lcdDisplay(0,3,DISP_CFONT|DISP_MEDIACY,"剩下%d次机会",TryCounter);
	lcdDisplay(0,5,DISP_CFONT|DISP_MEDIACY,"重新输入?");
	}
	
	while(1){
	retval = kbGetKey();
	if(retval == KEY_ENTER){
	return 0;
	}

	if(retval == KEY_CANCEL){
	errno = EMV_ERRNO_CANCEL;
	return -1;
	}
	}

8.6. emv_check_online_pin

Prototype	**int (*emv_check_online_pin)(void);**
Description	Enter the encipher online PIN. Main program keep the online pin by it self.
Parameter	NONE
Return Value	0: success -1: Failed If canceled Errno = EMV_ERRNO_CANCEL If BypassPIN Errno = EMV_ERRNO_BYPASS_PIN

Example	int emv_check_online_pin()
	{
	if (bypasspin)
	{
	errno = EMV_ERRNO_BYPASS_PIN;
	return -1;
	}
	else if (cancel)
	{
	errno = EMV_ERRNO_CANCEL;
	return -1;
	}
	Else
	{
	return 0;
	}
	}

8.7. emv_check_certificate

Prototype	`int (*emv_check_certificate)(void);`
Description	Enter the encipher online PIN. Main program keep the online pin by it self.
Parameter	NONE
Return Value	0: success -1: Failed

Example	`int emv_check_certificate(void)`
	`{`
	`  uint8_t CertType, Cert[41];`
	`  uint8_t CertLen;`
	`  int key;`
	
	`  if(emv_get_data_element("\x9F\x62",2,&CertLen,&CertType)){`
	`  errno = EMV_ERRNO_DATA;`
	`  return -1;`
	`  }`
	
	`  if(emv_get_data_element("\x9F\x61",2,&CertLen,Cert)){`
	`  errno = EMV_ERRNO_DATA;`
	`  return -1;`
	`  }`
	
	`  if (operator checked no problem)`
	`  return 0;`
	`  else`
	`  return -1;`
	`}`

8.8. emv_online_transaction_process

<table>
<tr><td>Prototype</td><td colspan="2">int (*emv_online_transaction_process)(uint8_t *RspCode,uint8_t *AuthCode, int *AuthCodeLen,
uint8_t *IAuthData, int *IAuthDataLen,
uint8_t *script, int *ScriptLen,
int *online_result)</td></tr>
<tr><td>Description</td><td colspan="2">Online Authenticate process.</td></tr>
<tr><td rowspan="8">Parameter</td><td>RespCode</td><td>Return ARC</td></tr>
<tr><td>AuthCode</td><td>Tag 89 from host</td></tr>
<tr><td>AuthCodeLen</td><td>Length of Value of Tag 89</td></tr>
<tr><td>IAuthData</td><td>Issuer Authenticate Data</td></tr>
<tr><td>IAuthDataLen</td><td>Issuer Authenticate data length</td></tr>
<tr><td>script</td><td>Issuer Script</td></tr>
<tr><td>ScriptLen</td><td>Issuer Script length.</td></tr>
<tr><td>Online_result</td><td>0: APPROVED
1: DECLIEN
2: Referal</td></tr>
<tr><td>Return Value</td><td colspan="2">0: success
-1: Failed
Errno = EMV_ERRNO_ONLINE_ERROR
EMV_ERRNO_ONLINE_TIMEOUT</td></tr>
<tr><td>Example</td><td colspan="2">int emv_online_transaction_process(uint8_t *RspCode,
uint8_t *AuthCode, int *AuthCodeLen,
uint8_t *IAuthData, int *IAuthDataLen,
uint8_t *script, int *ScriptLen,
int *online_result)
{
//Connect to the host and send request data
If no response ot connect failed
Errno = EMV_ERRNO_ONLINE_TIMEOUT;
Return -1;
Else if responsed data incorrect
Errno = EMV_ERRNO_ONLINE_ERROR;
Return -1;
Else</td></tr>
</table>

	{
	……
	Return 0;
	}
	
	}

8.9. emv_issuer_referral_process

Prototype	**int emv_issuer_referral_process (void)**	
Description	Issuer referral	
Parameter	NONE	
Return Value	0: APPROVED -1: Failed errno = EMV_ERRNO_DECLINE;	

Example	int emv_issuer_referral_process (void)
	{
	If Accept the transaction
	{
	Return 0;
	}
	Else
	{
	errno = EMV_ERRNO_DECLINE;
	return -1;
	}
	}	

8.10. emv_advice_process

Prototype	**int (*emv_advice_process)(int)**
Description	Issuer referral

Parameter	NONE	
Return Value	0: Success Other: Failed	

Example	int emv_advice_process(int Flag)
	{
	struct terminal_mck_configure config;
	int retval=0;
	
	emv_get_mck_configure(&config);
	
	if(!config.support_advices){
	return 0;
	}
	
	if(config.support_batch_data_capture){
	if(emv_check_force_accept()){
	//save the trans log and identify it's a FORCE ACCEPT transaction
	}else{
	//Offline Advice
	}
	}else{
	//Online Advice.
	}
	
	return retval;
	}	

8.11. emv_check_revocation_certificate

Prototype	**int (*emv_check_revocation_certificate)(uint8_t index,const void *rid,const void *cert_sn)**	
Description	Check the revocation list	
Parameter	index	CAPK Index
	rid	RID

	Cert_sn	Certificate serial number
Return Value	0: in the revocation list -1: not in the revocation list	

Example	int emv_check_crl(uint8_t index,const void *rid,const void *cert_sn)
	{
	int i = 0;
	
	if(!rid || !cert_sn){
	errno = EMV_ERRNO_INVAL;
	return -1;
	}
	
	for(i=0;i<gl_crl_num;i++){
	if((gl_crl[i].index == index)
	&& !memcmp((void *)gl_crl[i].rid,rid,5)
	&& !memcmp((void *)gl_crl[i].cert_sn,cert_sn,3)){
	return 0;
	}
	}
	errno = EMV_ERRNO_NODATA;
	return -1;
	}	

8.12. emv_check_exception_file

Prototype	**int emv_check_exception_file (uint8_t pan_len,const void *pan,uint8_t pan_sn)**	
Description	Check the exception file	
Parameter	Pan_len	Length of the PAN
	pan	PAN
	Pan_sn	PAN sequence number.

Return Value	0: in the exception list -1: not in the exception list

Example	int emv_check_except_file(uint8_t pan_len,const void *pan,uint8_t pan_sn)
	{
	int i=0;
	
	if(!pan || !pan_len|| (pan_len>10)){
	errno = EMV_ERRNO_INVAL;
	return -1;
	}
	
	for(i=0; i<gl_exception_file_num; i++){
	if((gl_exception_file[i].pan_len = pan_len)
	&& !memcmp((void *)gl_exception_file[i].pan, pan, pan_len)
	&& (gl_exception_file[i].pan_sn == pan_sn)){
	
	return 0;
	}
	}
	errno = EMV_ERRNO_NODATA;
	return -1;
	}

8.13. emv_get_transaction_log_amount.

Prototype	**int emv_get_transaction_log_amount(uint8_t pan_len,const void *pan,uint8_t pan_sn)**	
Description	Get transaction amount	
Parameter	Pan_len	Length of the PAN
	pan	PAN
	Pan_sn	PAN sequence number.
Return Value	-1: error >=0: Return count of the amount.	

Example	int emv_get_log_amount(uint8_t pan_len,const void *pan,uint8_t pan_sn)
	{
	int i=0;
	int amount=0;
	
	if(!pan || !pan_len || (pan_len>10)){
	errno = EMV_ERRNO_INVAL;
	return -1;
	}
	
	for(i=0; i<gl_translog_num; i++){
	if((gl_translog[i].pan_len == pan_len)
	&& !memcmp((uint8_t *)gl_translog[i].pan, pan, gl_translog[i].pan_len)
	&& (gl_translog[i].pan_sn == pan_sn)){
	amount += gl_translog[i].amount;
	}
	}
	
	return amount;
	}

8.14. emv_init_callback_func

Prototype	**int emv_init_callback_func(const struct emv_callback_func *ptFunc);**	
Description	Set callback function	
Parameter	ptFunc	Callback function structure
Return Value	0: Success Other: Failed	

Example	const struct emv_callback_func Func;
	memset(&Func, 0, sizeof(Func));
	Func.emv_candidate_apps_selection = emv_candidate_apps_selection;
	Func.emv_advice_process = emv_advice_process;
	Func.emv_check_certificate = emv_check_certificate;
	Func.emv_get_pin = emv_get_pin;
	……
	if (0 == emv_init_callback_func(&Func))
	{
	//Success
	}
	else
	{
	//Failed
	}

8.15. emv_get_callback_func

Prototype	**void emv_get_callback_func(struct emv_callback_func *ptFunc);**	
Description	Get callback function settings.	
Parameter	ptFunc	Callback function structure
Return Value	0: Success Other: Failed	

Example	const struct emv_callback_func Func;
	if (0 == emv_get_callback_func(&Func))
	{
	//Success
	}
	else
	{
	//Failed
	}	

9. Transaction Parameters

```
#define EMV_CASH        0x80
#define EMV_GOODS        0x40
#define EMV_SERVICE     0x20
#define EMV_CASHBACK   0x10
#define EMV_INQUIRY     0x08
#define EMV_TRANSFER    0x04
#define EMV_PAYMENT     0x02
#define EMV_ADMIN        0x01


struct emv_core_init{
    uint8_t terminal_id[8];
    uint8_t merchant_id[15];
    uint8_t merchant_cate_code[2];
    uint8_t merchant_name_loc_len;
    uint8_t merchant_name_loc[256];
    uint8_t transaction_type;
    uint8_t terminal_country_code[2];
    uint8_t transacion_currency_code[2];
    uint8_t refer_currency_code[2];
    uint8_t transacion_currency_exponent;
    uint8_t refer_currency_exponent;
    uint32_t refer_currency_coefficient;
};
```

9.1. emv_set_core_init_parameter

Prototype	**int emv_set_core_init_parameter(const struct emv_core_init *ptInit);**	
Description	Set parameters.	
Parameter	ptInit	Init parameters structure
Return Value	0: Success Other: Failed	

Example	struct emv_core_init Init;
	//initialize the structure data elements
	……
	if (0 == emv_set_core_init_parameter (&Init))
	{
	//Success
	}
	else
	{
	//Failed
	}

9.2. emv_get_core_init_parameter

Prototype	**int emv_get_core_init_parameter(struct emv_core_init *ptInit);**	
Description	Get parameters.	
Parameter	ptInit	Init parameters structure
Return Value	0: Success Other: Failed	

Example	struct emv_core_init Init;
	if (0 == emv_get_core_init_parameter (&Init))
	{
	//Success
	}
	else
	{
	//Failed
	}	

10. Transaction APIs

10.1. emv_get_all_candapps

```
struct candapp{
```

```
    uint8_t tCandAppName[33];  //Application Display Name
    uint8_t cFlgAPID;          //Application has Priority identify or not
    uint8_t cAPID;            //Application Priority Identify. If bit 8 is 1
then need confirm
};
```

Prototype	**int emv_get_all_candapps(uint8_t *appnum, struct candapp *applist);**	
Description	Get parameters.	
Parameter	appnum	Return Application count
	applist	Return Application list
Return Value	0: Success Other: Failed	

Example	uint8_t CandListNum=0;
	struct candapp applist[32];
	if (0 == emv_get_all_candapps (&CandListNum, applist))
	{
	  //Success
	}
	else
	{
	  //Failed
	}	

10.2. emv_application_select

Prototype	**int emv_application_select(int fd, uint32_t transno);**	
Description	Start a transaction, select application on the card.	
Parameter	fd	Card reader handle
	transno	Transaction sequence number.
Return Value	0: Success Other: Failed Error code reference the "Common Error Code Definition"	

Example	Fd = open iccard reader Transno++; emv_application_select(fd, transno);

10.3. emv_read_app_data

Prototype	**int emv_read_app_data(int fd);**	
Description	Read application data	
Parameter	fd	Card reader handle
Return Value	0: Success Other: Failed Error code reference the "Common Error Code Definition"	
Example		

10.4. emv_offline_data_authentication

Prototype	**int emv_offline_data_authentication(int fd);**	
Description	Offline data authentication	
Parameter	NONE	
Return Value	0: Success Other: Failed Error code reference the "Common Error Code Definition"	
Example		

10.5. emv_process_restriction

Prototype	**void emv_process_restriction(void);**	
Description	Process restriction	
Parameter	NONE	
Return Value	NONE	
Example		

10.6. emv_cardholder_verify

Prototype	**void emv_cardholder_verify (int fd);**	
Description	Cardholder Verify	
Parameter	fd	Smart card reader handle
Return Value	NONE	
Example		

10.7. emv_terminal_risk_manage

Prototype	**int emv_terminal_risk_manage(int fd);**	
Description	Terminal risk management	
Parameter	fd	Smart card reader handle
Return Value	NONE	

Example	

10.8. emv_terminal_action_analysis

Prototype	**int emv_terminal_action_analysis(int fd,int *need_online);**	
Description	Terminal action analyze	
Parameter	fd	Smart card reader handle
	Need_online	0: no need online 1: need online
Return Value	0: success Other: Failed	
Example		

10.9. emv_online_transaction

Prototype	**int emv_online_transaction(int fd);**	
Description	Online transaction. This function called after "emv_terminal_action_analyze()" online transaction needed.	
Parameter	fd	Smart card reader handle
Return Value	0: success Other: Failed	
Example		

10.10. emv_get_script_result

Prototype	**int emv_get_script_result(void *result, int *len);**	
Description	Get script result.	
Parameter	result	Buffer used to receive the result data
	len	Length of the script result.
Return Value	0: success Other: Failed	
Example		

10.11. emv_check_signature

Prototype	**int emv_check_signature(void);**
Description	Check the signature needed or not
Parameter	NONE
Return Value	0: No need Other: Need signature
Example	

10.12. emv_check_force_accept

```
int emv_check_force_accept(void);
```

Prototype	**int emv_check_force_accept(void);**
Description	Check the signature needed or not
Parameter	NONE

Return Value	0: not force accept Other: force accept
Example	

10.13. emv_check_online_flag

Prototype	**int emv_check_online_flag(void);**	
Description	Check the transaction is online or offline	
Parameter	None	
Return Value	0: offline Other: online	
Example		

10.14. emv_get_last_iccard_sw

Prototype	**uint16_t emv_get_last_iccard_sw(void);**	
Description	Get last Status Word of the SmartCard	
Parameter	None	
Return Value	Last Status word	
Example		

10.15. emv_application_select_no_gpo

Prototype	**int emv_application_select_no_gpo(int fd);**	
Description	Select application without GPO, this function used when need to read the transaction log etc.	
Parameter	fd	SmartCard reader handle
Return Value	0: success Other: failed	
Example		

10.16. emv_read_log_record

Prototype	**int emv_read_log_record(int fd,uint8_t record_no);**	
Description	Read Card transaction log	
Parameter	fd	SmartCard reader handle
	Record_no	Record Number.
Return Value	0: success Other: failed	
Example		

10.17. emv_get_log_item

Prototype	**int emv_get_log_item(const void *tag, uint8_t taglen, void *value, uint8_t *valuelen);**	
Description	Get log item data.	
Parameter	tag	Tag
	taglen	Length of the Tag

	value	Return value
	valuelen	Length of the value returned
Return Value	0: success Other: failed	
Example		

10.18. EMV Transaction Demo

```
Main()
{
    emv_set_kernel_type ();
    emv_init_callback_func()
    emv_init_data_element();
    emv_add_one_aid_info ();
    …….
    emv_add_one_ca_public_key ();
    emv_set_mck_configure ();
    emv_set_core_init_parameter();
    emv_application_select(icc_fd,transno);
     emv_read_app_data(icc_fd);
    emv_offline_data_authentication(icc_fd);
    emv_cardholder_verify(icc_fd);
    emv_terminal_risk_manage(icc_fd);
    emv_terminal_action_analysis(icc_fd,&need_online);
    if(need_online){
          retval = emv_online_transaction(icc_fd);
    }
    ……
    //print receipt, save the transaction log
}
```

10.19. EMV Read Card Transaction Log Demo

```
Main()
{
   int fd= -1;
```

```
    uint8_t LogEntry[10];
    uint8_t iLogEntry;
    uint8_t Reco;
    fd = open smartcard reader
    if (not powerup)
    {
        Return
    }
    If (0 ==  emv_application_select_no_gpo(fd))
    {
         if(0 != emv_check_data_element("\x9F\x4D",2))
         {
              Return;
         }
    }
    Else
    {
       Return;
    }
    iLogEntry = sizeof(LogEntry);
    memset(LogEntry, 0, sizeof(iLogEntry));
    emv_get_data_element("\x9F\x4D",2, & iLogEntry, LogEntry);
    if ((2 != iLogEntry) || (LogEntry[0]<(uint8_t)11) || (LogEntry[0] >
(uint8_t)20))
    {
       //Data incorrrect
       return;
    }
    Recno = 0;
    while((uint8_t)Reco <= LogEntry[1]){
    {
       If (0 == emv_read_log_record(icc_fd,Reco))
       {
           Emv_get_log_item() //get log item data.
           Emv_get_log_item() //get log item data.
       }
       Else
       {
           //Error
          Break;
       }
    }
}
```

11. PBOC APIs

11.1. emv_pboc_ec_enable

Prototype	**void emv_pboc_ec_enable(int mode);**	
Description	Disable/Enable PBOC EC	
Parameter	mode	0: disable 1: enable
Return Value	NONE	
Example		

11.2. emv_pboc_set_ec_ttl

Prototype	**int emv_pboc_set_ec_ttl(uint32_t value)**	
Function	Set the current EC terminal transaction limit	
Parameter	Value	EC limit value
Return Value	NONE	

11.3. emv_pboc_get_ec_ttl

Prototype	**uint32_t emv_pboc_get_ec_ttl(void);**
Function	Set the current EC terminal transaction limit
Parameter	None
Return Value	EC terminal transaction limit

11.4. emv_pboc_check_ec

Prototype	int emv_pboc_check_ec(void);
Function	Check the PBOC EC Enable/Disable
Parameter	None
Return Value	0: disable Other: Enabled

12. QPBOC&MSD APIs

12.1. Error Code Definition

```
#define EMV_QPBOC_OK                        0
#define EMV_QPBOC_TRANSACTION_DECLIEN         1
#define EMV_QPBOC_TRANSACTION_TERMINATED       2
#define EMV_QPBOC_APP_BLOCKED                3
#define EMV_QPBOC_CANCELED                     4
#define EMV_QPBOC_CONNECT_TRANSACTION_NEEDED   5
#define EMV_QPBOC_NEED_PBOC_TRANSACTION        6
#define EMV_QPBOC_
#define EMV_QPBOC_NEED_RETRY                  7

#define EMV_QPBOC_ICC_ERROR                    8
#define EMV_QPBOC_INVALID_DATA                9
#define EMV_QPBOC_DATA_MISSING                10
#define EMV_QPBOC_DATA_DUPLICATE               11
#define EMV_QPBOC_MEMORY_OVERFLOW            12
#define EMV_QPBOC_MEMORY_NO_ENOUGH              13
#define EMV_QPBOC_PROGRAMING_ERROR            14
#define EMV_QPBOC_COMMUNICATION_ERROR          15
#define EMV_QPBOC_CONDITIONS_NOT_SATISFIED     16
#define EMV_QPBOC_CARD_EXPIRED                17
#define EMV_QPBOC_CARD_IN_BLACK_LIST           18
```

12.2. Contactless Parameter

12.3. Contactless Parameter Structure

```
struct qpboc_parameters
{
    uint8_t   m_TransactionProperty[4];
    uint8_t  m_StatusCheckSupported;
};
```

12.4. emv_qpboc_set_parameter

Prototype	**int emv_qpboc_set_parameter(const struct qpboc_parameters * parameters);**
Function	Set QPBOC parameter
Parameter	Parameters: qpboc parameters
Return Value	Reference 14.1 Error Code Definition

12.5. emv_qpboc_get_parameter

Prototype	**int emv_qpboc_get_parameter(struct qpboc_parameters * parameters);**
Function	Get QPBOC parameter
Parameter	Parameters: return qpboc parameters
Return Value	Reference 14.1 Error Code Definition

12.6. emv_qpboc_pre_transaction

Prototype	**Int emv_qpboc_pre_transaction(uint32_t uiTarnsCounter, uint8_t TransType, uint32_t Amount);**	
Function	Retranslation	
Parameter	uiTransCounter	Transaction counter
	TransType	Transaction Type
	Amount	Transaction Amount

Return Value	Reference 14.1 Error Code Definition

12.7. emv_qpboc_read_data

Prototype	int emv_qpboc_read_data(int fd);	
Function	Read card data , notice to remove the smart from the reader after this function returned.	
Parameter	fd	Contact less reader handle
Return Value	Reference 14.1 Error Code Definition	

12.8. emv_qpboc_complete

Prototype	int emv_qpboc_complete(void);	
Function	Complete the transaction	
Parameter	NONE	
Return Value	Reference 14.1 Error Code Definition	

12.9. emv_qpboc_signature_needed

Prototype	int emv_qpboc_signature_needed(int * pSignature);	
Function	Check is the signature needed.	
Parameter	pSignature	0: no need Other: needed
Return Value	Reference 14.1 Error Code Definition	

12.10. emv_qpboc_get_app_type

```
typedef enum
{
    EMV_MSD,
        EMV_QPBOC,
        EMV_PBOC
}EMV_APP_TYPE;
```

```
typedef EMV_APP_TYPE * LP_EMV_APP_TYPE;
```

Prototype	**int emv_qpboc_get_app_type(LP_EMV_APP_TYPE pAppType);**	
Function	Get the current application type	
Parameter	pAppType	EMV_MSD, EMV_QPBOC, EMV_PBOC
Return Value	Reference 14.1 Error Code Definition	

12.11. emv_qpboc_select_no_gpo

Prototype	**int emv_qpboc_select_no_gpo(int fd);**	
Function	Select the application without GPO(with PPSE)	
Parameter	fd	Smart Card reader handle
Return Value	Reference 14.1 Error Code Definition	

12.12. emv_qpboc_need_save_failed_log

Prototype	**int emv_qpboc_need_save_failed_log(void);**
Function	Check is the log need to save
Parameter	NONE
Return Value	0: no need Other: need to save.

12.13. QPBOC&MSD Demo

```
int Main()
{
    emv_qpboc_set_parameter();
    emv_qpboc_pre_transaction();

    detect contactless card

    emv_qpboc_read_data();
    notify to remove the smartcard
```

```
    emv_qpboc_complete();
    print receipt
    return 0;
}
```

13. VISA APIs

13.1. Head file visa.h

```
#ifndef VISA_H_
#define VISA_H_
#include "emvapi.h"

#define VISA_OK                                 0
#define VISA_TRANSACTION_DECLIEN                1
#define VISA_TRANSACTION_TERMINATED             2
#define VISA_APP_BLOCKED                        3
#define VISA_CANCELED                           4
#define VISA_CONTACT_TRANSACTION_NEEDED         5
#define VISA_NEED_VSDC_TRANSACTION              6
#define VISA_NEED_RETRY                         7

#define VISA_ICC_ERROR                          8
#define VISA_INVALID_DATA                       9
#define VISA_DATA_MISSING                       10
#define VISA_DATA_DUPLICATE                     11
#define VISA_MEMORY_OVERFLOW                    12
#define VISA_MEMORY_NO_ENOUGH                   13
#define VISA_PROGRAMING_ERROR                   14
#define VISA_COMMUNICATION_ERROR                15
#define VISA_CONDITIONS_NOT_SATISFIED           16
#define VISA_CARD_EXPIRED                       17
#define VISA_CARD_IN_BLACK_LIST                 18
#define VISA_CANNOT_CONNECT_DECLINE             19
#define VISA_NO_APP                             20
#define VISA_ISSUER_UPDATE_FAILED               21
#define VISA_NEED_RE_SHOW_CARD                  22
#define VISA_APDU_TIMEOUT                       23

#define VISA_ENABLED                            1
#define VISA_DISABLED                           0
```

```
#define VISA_ZERO_CHECK_DISABLE		VISA_DISABLED
#define VISA_ZERO_CHECK_OPTION1		1
#define VISA_ZERO_CHECK_OPTION2		2

#pragma pack(1)
struct visa_parameters
{
	uint8_t	m_TransactionProperty[4];
	uint8_t		m_StatusCheck;
	uint8_t		m_ZeroCheck;
	uint8_t		m_TransactionLimitCheck;
	uint8_t		m_FloorLimitCheck;
	uint8_t		m_CVMRequireCheck;
	uint32_t	m_Transaction_Limit;
	uint32_t		m_Floor_Limit;
	uint32_t	m_CVM_Require_Limit;
	uint8_t		m_CVN17_Supported;
	uint8_t		m_Track1_Supported;
	uint8_t		m_Track2_Supported;
};
#pragma pack()

typedef enum
{
	VISA_UNKNOW_APP,
	VISA_MSD_CVN17,
		VISA_MSD_LEGACY,
		VISA_QVSDC
}VISA_APP_TYPE;

typedef VISA_APP_TYPE * LP_VISA_APP_TYPE;

#define VISA_TRANS_PURCHASE 0x00
#define VISA_TRANS_CASH		0x01
#define VISA_TRANS_REFUND	0x20


struct visa_terminal_aid
{
	uint8_t aid_len;
	uint8_t aid[16];
	uint8_t support_partial_aid_select;
	uint8_t application_priority;
```

```
    uint32_t target_percentage;
    uint32_t maximum_target_percentage;
    uint32_t terminal_floor_limit;
    uint32_t threshold_value;
    uint8_t terminal_action_code_denial[5];
    uint8_t terminal_action_code_online[5];
    uint8_t terminal_action_code_default[5];
    uint8_t acquirer_identifier[6];
    uint8_t len_of_default_ddol;
    uint8_t default_ddol[254];
    uint8_t len_of_default_tdol;
    uint8_t default_tdol[254];
    uint8_t application_version[2];
    uint8_t len_of_terminal_risk_management_data;
    uint8_t terminal_risk_management_data[8];

    struct visa_parameters m_cl_normal_param;     //Default Parameters
    struct visa_parameters m_cl_cash_param;       //Cash transaction Prarmeters
    struct visa_parameters m_cl_cashback_param; //Cashback parameters
};


struct visa_drl_item{
    uint8_t program_id_len;
    uint8_t program_id[16];
    struct visa_parameters m_cl_normal_param; //Default parameters
    struct visa_parameters m_cl_cash_param;     //Cash transaction parameters
    struct visa_parameters m_cl_cashback_param; //Cashback transaction parameters
};



void visa_drl_enable(void);
void visa_drl_disable(void);
int   visa_drl_status(void);
void visa_drl_clear(void);
int   visa_drl_add(const struct visa_drl_item * drl);
int   visa_drl_delete(uint8_t program_id_len, const uint8_t * program_id);
int   visa_drl_count(void);
int   visa_drl_get(uint32_t index, struct visa_drl_item * drl);



void visa_terminal_aid_clear(void);
```

```
int    visa_terminal_aid_add(const struct visa_terminal_aid *info);
int    visa_terminal_aid_get(int index, struct visa_terminal_aid *info);
int    visa_terminal_aid_delete(uint8_t aid_len, const void *aid);
int    visa_terminal_aid_count(void);
int    visa_terminal_aid_check(uint8_t aid_len, const void *aid);


typedef int (*fn_visa_detect_broken)(unsigned int DetectHandle); //If broken detected, return none zero
void visa_set_broken_detect_func(fn_visa_detect_broken    pFn_DetectBroken, unsigned int DetectHandle);
int visa_get_broken_source(void);

int visa_get_parameter(struct visa_parameters * parameters);

int visa_pre_transaction(uint32_t uiTarnsCounter, uint8_t TransType, uint32_t Amount, uint32_t
AmountOther, uint8_t withcashback);
int visa_transaction(int fd);
int visa_complete(void);
int visa_issuer_update_need(void);
int visa_issuer_update(int fd);
int visa_signature_needed(int * pSignature);
int visa_get_app_type(LP_VISA_APP_TYPE pAppType);
int visa_need_save_failed_log(void);
int visa_msd_get_track1(char * pTrack1, unsigned int uiSize);
int visa_msd_get_track2(char * pTrack2, unsigned int uiSize);
int visa_get_terminal_entry_capability(char * pszEntryCapability, unsigned int uiSize);
#endif
```

13.2. Dynamic Parameter

13.2.1. visa_drl_enable

Prototype	void visa_drl_enable(void);
Function	Enable the dynamic parameter
Parameter	None
Return Value	None

13.2.2. visa_drl_disable

Prototype	void visa_drl_disable(void);

Function	Disable the dynamic parameter
Parameter	None
Return Value	None

13.2.3. visa_drl_clear

Prototype	void visa_drl_clear(void);
Function	Clear all of the dynamic parameter
Parameter	None
Return Value	None

13.2.4. visa_drl_add

Prototype	void visa_drl_add(const struct visa_drl_item * drl);
Function	Add a Dynamic parameter to the kernel
Parameter	Drl: the paramter to load to the kernel
Return Value	0: success, other: failed

13.2.5. visa_drl_delete

Prototype	void visa_drl_delete(uint8_t program_id_len, const uint8_t * program_id);
Function	Delete a Dynamic parameter from the kernel
Parameter	Program_id_len: program id length Program_id : program id value
Return Value	0: success, other: failed

13.2.6. visa_drl_count

Prototype	int visa_drl_count(void);
Function	Get the drl count from the kernel
Parameter	None
Return Value	>=0 success, return the count of the drl. <0 failed.

13.2.7. visa_drl_get

Prototype	int visa_drl_get(uint32_t index, struct visa_drl_item * drl);
Function	Get a drl item from the kernel
Parameter	Index : index of the drl. Value range 0 to the drl count -1 drl : drl item value
Return Value	0 success other failed.

13.3. VISA Terminal Application

13.3.1. fn_callback_visa_terminal_aid_count

Prototype	typedef int (*fn_callback_visa_terminal_aid_count)(void);	
Description	Retrieve the terminal AID count	
Parameters	NONE	
Return Value	0: success other: Failed	

Example	int callback_visa_terminal_aid_count (void)
	{
	return 10;
	}	

13.3.2. fn_callback_visa_terminal_aid_get

Prototype	typedef int (*fn_callback_visa_terminal_aid_get)(int index, void * pAidInfo, unsigned int *puiSize);	
Description	Retrieve the terminal AID by the index	
Parameters	index	Terminal AID Storage index
	pAidInfo	Buffer used to save the AID information
	puiSize	Pass in the buffer size, pass out the size of the AID information.
Return Value	0: success other: Failed	

Example	int callback_visa_terminal_aid_get (int index, void * pAidInfo, unsigned int *puiSize);
	{
	int retval = 0;
	memcpy(pAidInfo, Aidinfo, sizeof(*AidInfo));
	*puiSize = sizeof(*AidInfo);
	return retval;
	}

13.3.3. visa_aid_manage_set_callback

Prototype	**void visa_aid_manage_set_callback(fn_callback_visa_terminal_aid_count callback_visa_terminal_aid_count, fn_callback_visa_terminal_aid_get callback_visa_terminal_aid_get);**	
Description	Set AID callback function.	
Parameters	callback_visa_terminal_aid_count	Callback function pointer, this function will return number of the aid count.
	callback_visa_terminal_aid_get	Callback function pointer, this function will used to retrieve a specified index of the terminal AID.
Return Value	NONE	
Example	visa_aid_manage_set_callback (callback_visa_terminal_aid_count, callback_visa_terminal_aid_get);	

13.3.4. visa_terminal_aid_clear

Prototype	void visa_terminal_aid_clear(void);
Function	Clear all aid from the kernel
Parameter	None
Return Value	0 success other failed.

13.3.5. visa_terminal_aid_add

Prototype	void visa_terminal_aid_add(void);
Function	Add a terminal aid to the kernel
Parameter	Info: the aid information want to load to the kernel
Return Value	0 success other failed.

13.3.6. visa_terminal_aid_add

Prototype	void visa_terminal_aid_add(const struct visa_terminal_aid *info);
Function	Add a terminal aid to the kernel
Parameter	Info: the aid information want to load to the kernel
Return Value	0 success other failed.

13.3.7. visa_terminal_aid_get

Prototype	void visa_terminal_aid_get(struct visa_terminal_aid *info);
Function	Get a terminal aid information from the kernel
Parameter	Info: the aid informationl
Return Value	0 success other failed.

13.3.8. visa_terminal_aid_delete

Prototype	void visa_terminal_aid_delete(uint8_t aid_len, const void *aid);
Function	Delete the aid
Parameter	Len: the length of the aid Aid: aid value
Return Value	0 success other failed.

13.3.9. visa_terminal_aid_count

Prototype	void visa_terminal_aid_count(void);
Function	Get the terminal aid count from the kernel
Parameter	None
Return Value	>=0 success, return the aid count. other failed.

13.3.10. visa_terminal_aid_check

Prototype	void visa_terminal_aid_check(uint8_t aid_len, const void *aid);
Function	Check is the aid in the kernel already.
Parameter	Len: the length of the aid Aid: aid value
Return Value	0 Aid exists other Don't exists

13.4. Transaction

13.4.1. visa_set_broken_detect_func

Prototype	oid visa_set_broken_detect_func(fn_visa_detect_broken pFn_DetectBroken, unsigned int DetectHandle);
Function	Set the broken detect callback function

Parameter	pFn_detectBroken:
	typedef int (*fn_visa_detect_broken)(unsigned int DetectHandle); //If broken detected, return none zero
	DetectHandle: the callback handle

Return Value	0 success other failed

13.4.2. visa_get_broken_source

Prototype	Int visa_get_broken_source(void)
Function	Get the broken source id
Parameter	If visa_transaction return VISA_TRANSACTION_TERMINATED, then call this function to get the broken source. If zero means don't have any broken source.
Return Value	Get the broken source

13.4.3. visa_get_parameter

Prototype	int visa_get_parameter(struct visa_parameters * parameters);
Function	
Parameter	If visa_transaction return VISA_TRANSACTION_TERMINATED, then call this function to get the broken source
Return Value	0 success Other failed.

13.4.4. visa_pre_transaction

Prototype	int visa_pre_transaction(uint32_t uiTarnsCounter, uint8_t TransType, uint32_t Amount, uint32_t AmountOther, uint8_t withcashback);
Function	Pass the transaction information to the kernel
Parameter	uiTransCounter: terminal transaction counter TransType : Transaction Type VISA_TRANS_PURCHASE 0x00 VISA_TRANS_CASH 0x01 VISA_TRANS_REFUND 0x20 Amount : Amount AmountOther : cashback amount Withcashback : 1 Is a cash back transaction. Variable when TransType is VISA_TRANS_PURCHASE. 0 not a cash back transaction

Return Value	0 success Other failed.

13.4.5. visa_transaction

Prototype	int visa_transaction(int fd);
Function	Start to communicate with the card.
Parameter	Fd: the contactless reader handle. NOTICT: Must et the global value __ICCARD_EXCHANGE_APDU before this function calling. Must power on the contactless before this function calling. Close the contactless reader after this function called.
Return Value	0 success Other failed.

13.4.6. visa_complete

Prototype	int visa_complete(void);
Function	Complete the transaction offline or online.
Parameter	None
Return Value	0 success Other failed.

13.4.7. visa_signature_needed

Prototype	int visa_signature_needed(int * pSignature);
Function	Check is the signature needed.
Parameter	pSignature: 0 don't need signature Other need signature
Return Value	0 success Other failed.

13.4.8. visa_get_app_type

Prototype	int visa_get_app_type(LP_VISA_APP_TYPE pAppType);
Function	Get the application type of the current transaction
Parameter	pAppType: Application type VISA_UNKNOW_APP, VISA_MSD_CVN17, VISA_MSD_LEGACY, VISA_QVSDC
Return Value	0 success Other failed.

13.4.9. visa_need_save_failed

Prototype	int visa_need_save_failed_log(void);
Function	Check is the transaction log need to save to log file. 1. If last APDU of the when visa_transaction to the card timeout. 2. If FDDA failed and offline declined
Parameter	None
Return Value	0 Need save Other no need

13.4.10. visa_get_terminal_entry_capability

Prototype	int visa_get_terminal_entry_capability(char * pszEntryCapability, unsigned int uiSize);
Function	Get terminal entry capability
Parameter	pszEntrycapability: buffer to save the the Entry capability uiSize : buffer size
Return Value	0 Success Other Failed

13.5. Issuer Update

13.5.1. visa_issuer_update_need

Prototype	**int** visa_issuer_update_need(void)
Function	Check is the issuer update needed.
Parameter	None
Return Value	0 need to do issuer update Other no need

13.5.2. visa_issuer_update_need

Prototype	**int** visa_issuer_update_need(int fd)
Function	Start to do issuer update. Fd: the contactless reader handle. NOTICT: Must et the global value __ICCARD_EXCHANGE_APDU before this function calling. Must power on the contactless before this function calling. Close the contactless reader after this function called.
Parameter	None
Return Value	0 success Other failed.

13.6. MSD Track Data Generation

13.6.1. visa_msd_get_track1

Prototype	int visa_msd_get_track1(char * pTrack1, unsigned int uiSize);
Function	Generate the track1 data
Parameter	pTrack1: buffer to save the track1 data uiSize : buffer size
Return Value	0 Success Other Failed

13.6.2. visa_msd_get_track2

Prototype	int visa_msd_get_track2(char * pTrack2, unsigned int uiSize);
Function	Generate the track2 data
Parameter	pTrack2: buffer to save the track1 data uiSize : buffer size
Return Value	0 Success Other Failed

13.7. Example of the Transaction

```
int main(){
    emv_set_kernel_type(KERNEL_TYPE_VISA);
    emv_init_callback_func();
    emv_init_data_element();
    emv_set_mck_configure();
    emv_set_core_init_parameter()
    emv_delete_all_ca_public_keys();

    emv_add_one_ca_public_key();
    emv_add_one_ca_public_key();

    visa_terminal_aid_clear();
    visa_terminal_aid_add();

    visa_drl_enable() or visa_drl_disable()
    visa_drl_clear();
    visa_drl_add();
    Choise transaction type
    Enter transaction Amount, Cash back amount
    visa_pre_transaction();

    Open the contactless readeer
    Power on the card

    visa_transaction()
    As to remove the card
    Close the contactless reader
    visa_complete()
```

```
    If (0 == visa_issuer_update_need()){
        Open the contactless readeer
        Power on the card

        visa_issuer_update();
        As to remove the card
        Close the contactless reader
    }

    Print Receipt
    Show the transaction status


    return 0;
}
```

14. EMV Contactless API

14.1. Head File emv_cl.h

```
#define EMV_CL_OK_APPROVED          0
#define EMV_CL_DECLINE              1
#define EMV_CL_NEED_ONLINE          2
#define EMV_CL_SELECT_NEXT          3
#define EMV_CL_OTHER_INTERFACE      4
#define EMV_CL_OTHER_CARD           5
#define EMV_CL_TRY_AGAIN            6
#define EMV_CL_NEED_ANOTHER_TAP     7
#define EMV_CL_TERMINATED           8
#define EMV_CL_ERROR                9
#define EMV_CL_NO_SUPPORTED         10
#define EMV_CL_CANCELED             11


#define EMV_CL_PURCHASE                     0x00
#define EMV_CL_PURCHASE_WITH_CASHBACK       0x09
#define EMV_CL_CASH_WITHDRAWVAL             0x01
#define EMV_CL_CASH_DISBURSEMENT            0x17
#define EMV_CL_REFUND                       0x20
```

```
typedef unsigned char EMV_CL_TRANS_TYPE;

typedef    enum{
      EMV_CL_APP_UNKNOW,
            EMV_CL_APP_MAGSTRIPE,
            EMV_CL_APP_EMV
}EMV_CL_APP_TYPE;

typedef    enum{
            EMV_CL_KERNEL_UNKNOW,
            EMV_CL_KERNEL_1,
            EMV_CL_KERNEL_2,
            EMV_CL_KERNEL_3,
            EMV_CL_KERNEL_4,
            EMV_CL_KERNEL_5,
            EMV_CL_KERNEL_6,
            EMV_CL_KERNEL_7
}EMV_CL_KERNEL_ID;



#pragma pack(1)
typedef struct {
      EMV_CL_TRANS_TYPE      trans_type;
      EMV_CL_KERNEL_ID       kernel_id;
      int                          supported;
      int                           aid_len;
      unsigned char        aid[16];
      unsigned int         param_len;
      unsigned char        param[1];
}emv_cl_kernel_param;
#pragma pack()


typedef enum{
      EMV_CL_APDU_PROCESSING,
            EMV_CL_APDU_ABORTED,
            EMV_CL_APDU_DONE
}EMV_CL_APDU_STATUS;
```

```
typedef int (*fn_emv_cl_apdu_send)(int fd, unsigned int sendlen, const void * senddata);
typedef int (*fn_emv_cl_apdu_resp_get)(int fd, int AllowStop, unsigned int *recvlen, void *recvdata,
EMV_CL_APDU_STATUS * status);


void emv_cl_reader_set_callback(fn_emv_cl_apdu_send                apdu_send,
                                fn_emv_cl_apdu_resp_get        apdu_resp_get);


typedef int (*fn_emv_cl_kernel_param_get_count)(void);
typedef int (*fn_emv_cl_kernel_param_get)(int index,
                                          void * pBuff,
                                          unsigned int * BuffSize);



void emv_cl_kernel_param_set_callback(fn_emv_cl_kernel_param_get_count param_get_count,
                                      fn_emv_cl_kernel_param_get           param_get);

int emv_cl_kernel_param_init(void);
int emv_cl_kernel_param_add(const emv_cl_kernel_param * config);
int emv_cl_kernel_param_remove(
                                    EMV_CL_TRANS_TYPE    trans_type,
                                    EMV_CL_KERNEL_ID     KernelID,
                                    const unsigned char *        aid,
                                    unsigned int                 aid_len);
int emv_cl_kernel_param_get_count(void);
int emv_cl_kernel_param_get(int index,
                            emv_cl_kernel_param ** config);


int emv_cl_trans_param_set(EMV_CL_KERNEL_ID KernelID,
                           const unsigned char * param,
                           unsigned int             param_len);

int emv_cl_trans_param_get(EMV_CL_KERNEL_ID KernelID,
                           unsigned char            * param,
                           unsigned int             * param_len);


int emv_cl_pre_processing(void);
int emv_cl_entry_point( int ifd, int select_next,
                        EMV_CL_KERNEL_ID *     pKernelID,
```

unsigned char * aid, unsigned int * aid_len,
unsigned char * FCI, unsigned int * FCI_Len
);

int emv_cl_transaction(int ifd,
EMV_CL_KERNEL_ID KernelID,
unsigned char * aid, unsigned int aid_len,
unsigned char * FCI, unsigned int FCI_Len);
int emv_cl_transaction_complete(EMV_CL_KERNEL_ID KernelID);

int emv_cl_torn_record_delete_all(EMV_CL_KERNEL_ID KernelID);

14.2. Contactless Reader Functions

14.2.1. fn_emv_cl_apdu_send

Prototype	typedef int (*fn_emv_cl_apdu_send)(int fd, unsigned int sendlen, const void * senddata);
Function	Callback function, used to send APDU command to the card.
Parameter	fd : reader handle sendlen : APDU Length senddata: APDU command buffer.
Return Value	0 : Success <0: Failed

14.2.2. fn_emv_cl_apdu_resp_get

Prototype	typedef int (*fn_emv_cl_apdu_resp_get)(int fd, int AllowStop, unsigned int *recvlen, void *recvdata, EMV_CL_APDU_STATUS * status);
Function	Callback function, used to get APDU respose from the card.
Parameter	fd : reader handle recvlen: received data Length recvdata: Data buffer to save the received response data. Status : EMV_CL_APDU_PROCESSING EMV_CL_APDU_ABORTED,

	EMV_CL_APDU_DONE
Return Value	0 : Success <0: Failed

14.2.3. fn_emv_cl_apdu_resp_get

Prototype	void emv_cl_reader_set_callback(fn_emv_cl_apdu_send apdu_send, fn_emv_cl_apdu_resp_get apdu_resp_get);
Function	Set the contactless reader callback function
Parameter	apdu_send : send APDU callback function apdu_resp_get: Get APDU response from the card
Return Value	NONE

14.3. Contactless kernel parameter

```
#pragma pack(1)
typedef struct {
    EMV_CL_TRANS_TYPE    trans_type;   //Transaction Type
    EMV_CL_KERNEL_ID     kernel_id;    //Kernel ID
    int                  supported;    // 0: No support   1: Support
    int                  aid_len;      //Length of the AID
    unsigned char        aid[16];      //AID
    unsigned int         param_len; //Kernel Parameter length
    unsigned char        param[1];     //Param
}emv_cl_kernel_param;
#pragma pack()
```

For paypass, kernel parameter is TLV formatted binary data.

14.3.1. fn_emv_cl_kernel_param_get_count

Prototype	typedef int (*fn_emv_cl_kernel_param_get_count)(void);
Function	Return count of the contactless kernel params.
Parameter	NONE
Return Value	int

14.3.2. fn_emv_cl_kernel_param_get

Prototype	typedef int (*fn_emv_cl_kernel_param_get)(int index, void * pBuff, unsigned int * BuffSize);
Function	Get a kernel parameter by index
Parameter	NONE
Return Value	0 : success Other : Failed

14.3.3. fn_emv_cl_kernel_param_get

Prototype	void emv_cl_kernel_param_set_callback(fn_emv_cl_kernel_param_get_count param_get_count, fn_emv_cl_kernel_param_get param_get);
Function	Set callback function
Parameter	param_get_count: callback function, return the count of the parameters. param_get : Get a kernel parameter by index.
Return Value	none

14.3.4. emv_cl_kernel_param_init

Prototype	int emv_cl_kernel_param_init(void)
Function	Initialize the parameters.
Parameter	NONE
Return Value	NONE

14.3.5. emv_cl_kernel_param_add

Prototype	int emv_cl_kernel_param_add(const emv_cl_kernel_param * config)
Function	Add a parameter.
Parameter	Config: Kernel configuration parameter.

Return Value	EMV_CL_OK_APPROVED EMV_CL_DECLINE EMV_CL_NEED_ONLINE EMV_CL_SELECT_NEXT EMV_CL_OTHER_INTERFACE EMV_CL_OTHER_CARD EMV_CL_TRY_AGAIN EMV_CL_NEED_ANOTHER_TAP EMV_CL_TERMINATED EMV_CL_ERROR EMV_CL_NO_SUPPORTED EMV_CL_CANCELED

14.3.6. emv_cl_kernel_param_remove

Prototype	int emv_cl_kernel_param_remove(EMV_CL_TRANS_TYPE trans_type, EMV_CL_KERNEL_ID KernelID, const unsigned char * aid, unsigned int aid_len)
Function	Remove a kernel parameter from the kernel library
Parameter	Trans_type: transaction type KernelID : Kernel ID Aid : Aid Aid_len : Length of the AID
Return Value	EMV_CL_OK_APPROVED EMV_CL_DECLINE EMV_CL_NEED_ONLINE EMV_CL_SELECT_NEXT EMV_CL_OTHER_INTERFACE EMV_CL_OTHER_CARD EMV_CL_TRY_AGAIN EMV_CL_NEED_ANOTHER_TAP EMV_CL_TERMINATED EMV_CL_ERROR EMV_CL_NO_SUPPORTED EMV_CL_CANCELED

14.3.7. emv_cl_kernel_param_get_count

Prototype	int emv_cl_kernel_param_get_count(void)
Function	Get the count of the kernel parameters.
Parameter	NONE
Return Value	Count of the kernel parameters.

14.3.8. emv_cl_kernel_param_get

Prototype	int emv_cl_kernel_param_get(int index, emv_cl_kernel_param ** config)
Function	Get a kernel configuration parameter from the kernel by the index.
Parameter	Index: index of the parameter Config: Buffer to save the Kernel configuration parameter.
Return Value	EMV_CL_OK_APPROVED EMV_CL_DECLINE EMV_CL_NEED_ONLINE EMV_CL_SELECT_NEXT EMV_CL_OTHER_INTERFACE EMV_CL_OTHER_CARD EMV_CL_TRY_AGAIN EMV_CL_NEED_ANOTHER_TAP EMV_CL_TERMINATED EMV_CL_ERROR EMV_CL_NO_SUPPORTED EMV_CL_CANCELED

14.3.9. emv_cl_trans_param_set

Prototype	int emv_cl_trans_param_set(EMV_CL_KERNEL_ID KernelID, const unsigned char * param, unsigned int param_len);
Function	Set the traction parameters from the specified kernel id.
Parameter	KernelID, :Kernel ID Param : Buffer to save the kernel parameter

	param_len : Length of the parameter.
Return Value	EMV_CL_OK_APPROVED EMV_CL_DECLINE EMV_CL_NEED_ONLINE EMV_CL_SELECT_NEXT EMV_CL_OTHER_INTERFACE EMV_CL_OTHER_CARD EMV_CL_TRY_AGAIN EMV_CL_NEED_ANOTHER_TAP EMV_CL_TERMINATED EMV_CL_ERROR EMV_CL_NO_SUPPORTED EMV_CL_CANCELED

14.3.10. emv_cl_trans_param_get

Prototype	int emv_cl_trans_param_get(EMV_CL_KERNEL_ID KernelID, unsigned char * param, unsigned int * param_len);
Function	Get the traction parameters from the specified kernel id.
Parameter	KernelID, :Kernel ID Param : Buffer to save the kernel parameter param_len : Length of the parameter.
Return Value	EMV_CL_OK_APPROVED EMV_CL_DECLINE EMV_CL_NEED_ONLINE EMV_CL_SELECT_NEXT EMV_CL_OTHER_INTERFACE EMV_CL_OTHER_CARD EMV_CL_TRY_AGAIN EMV_CL_NEED_ANOTHER_TAP EMV_CL_TERMINATED EMV_CL_ERROR EMV_CL_NO_SUPPORTED EMV_CL_CANCELED

14.4. Contactless Transaction Functions

14.4.1. emv_cl_pre_processing

Prototype	int emv_cl_pre_processing(void);
Function	Preprocessing. Set the TTQ etc.
Parameter	NONE
Return Value	EMV_CL_OK_APPROVED EMV_CL_DECLINE EMV_CL_NEED_ONLINE EMV_CL_SELECT_NEXT EMV_CL_OTHER_INTERFACE EMV_CL_OTHER_CARD EMV_CL_TRY_AGAIN EMV_CL_NEED_ANOTHER_TAP EMV_CL_TERMINATED EMV_CL_ERROR EMV_CL_NO_SUPPORTED EMV_CL_CANCELED

14.4.2. emv_cl_entry_point

Prototype	int emv_cl_entry_point(int ifd, int select_next, EMV_CL_KERNEL_ID * pKernelID, unsigned char * aid, unsigned int * aid_len, unsigned char * FCI, unsigned int * FCI_Len);
Function	EMV Contactless Entry point
Parameter	Ifd : handle if the contactless reader KernelID: Kernel ID Aid : application aid Aid_len: length of the aid. FCI : FCI FCI_Len: Length of the FCI
Return Value	EMV_CL_OK_APPROVED EMV_CL_DECLINE

	EMV_CL_NEED_ONLINE EMV_CL_SELECT_NEXT EMV_CL_OTHER_INTERFACE EMV_CL_OTHER_CARD EMV_CL_TRY_AGAIN EMV_CL_NEED_ANOTHER_TAP EMV_CL_TERMINATED EMV_CL_ERROR EMV_CL_NO_SUPPORTED EMV_CL_CANCELED

14.4.3. emv_cl_transaction

Prototype	int emv_cl_transaction(int ifd, EMV_CL_KERNEL_ID KernelID, unsigned char * aid, unsigned int aid_len, unsigned char * FCI, unsigned int FCI_Len);
Function	Phrase the FCI, and read data from the card.
Parameter	Ifd : handle if the contactless reader KernelID: Kernel ID Aid : application aid Aid_len: length of the aid. FCI : FCI FCI_Len: Length of the FCI
Return Value	EMV_CL_OK_APPROVED EMV_CL_DECLINE EMV_CL_NEED_ONLINE EMV_CL_SELECT_NEXT EMV_CL_OTHER_INTERFACE EMV_CL_OTHER_CARD EMV_CL_TRY_AGAIN EMV_CL_NEED_ANOTHER_TAP EMV_CL_TERMINATED EMV_CL_ERROR EMV_CL_NO_SUPPORTED EMV_CL_CANCELED

14.4.4. emv_cl_transaction_complete

Prototype	int emv_cl_transaction_complete(EMV_CL_KERNEL_ID KernelID);
Function	Complete the transaction, called after emv_cl_transaction
Parameter	KernelID: Kernel ID
Return Value	EMV_CL_OK_APPROVED EMV_CL_DECLINE EMV_CL_NEED_ONLINE EMV_CL_SELECT_NEXT EMV_CL_OTHER_INTERFACE EMV_CL_OTHER_CARD EMV_CL_TRY_AGAIN EMV_CL_NEED_ANOTHER_TAP EMV_CL_TERMINATED EMV_CL_ERROR EMV_CL_NO_SUPPORTED EMV_CL_CANCELED

14.4.5. emv_cl_torn_record_delete_all

Prototype	int emv_cl_torn_record_delete_all(EMV_CL_KERNEL_ID KernelID);
Function	Delete all the torn record data from kernel specified by the Kernel ID
Parameter	KernelID: Kernel ID
Return Value	EMV_CL_OK_APPROVED EMV_CL_DECLINE EMV_CL_NEED_ONLINE EMV_CL_SELECT_NEXT EMV_CL_OTHER_INTERFACE EMV_CL_OTHER_CARD EMV_CL_TRY_AGAIN EMV_CL_NEED_ANOTHER_TAP EMV_CL_TERMINATED EMV_CL_ERROR EMV_CL_NO_SUPPORTED EMV_CL_CANCELED

15. Paypass API

```
typedef struct{
    int                 m_amount_present;                   //1: Present 0: Not present
    unsigned int        m_amount;                           // Amount
    int                 m_amount_other_present;             //1: Present 0:Not present
    unsigned int        m_amount_other;                     // Amount Other
    int                 m_transaction_type_present;         //1: Present 0:Not present
    EMV_CL_TRANS_TYPE   m_transaction_type;                 // Transaction Type
    int                 m_transaction_catagory_code_present;//1: Present 0:Not present
    unsigned char       m_transaction_catagory_code;        //Transaction category code
    int                 m_transaction_currency_code_present;//1: Present 0:Not present
    unsigned char       m_transaction_currency_code[2];     //Transaction currency code
    unsigned char       m_transaction_currency_exponent;    //Transaction currency exponent
    int                 m_balance_read_before_gac_present;  //1: Present 0:Not present
    int                 m_balance_read_before_gac_len;      //Balance read before GAC len
    unsigned char       m_balance_read_before_gac[6];       //Balance read before GAC
    int                 m_balance_read_after_gac_present;   //1: Present 0:Not present
    int                 m_balance_read_after_gac_len;       //Length of balance read after GAC
    unsigned char       m_balance_read_after_gac[6];        //Balance Read after GAC
    unsigned char       m_merchant_custom_data[20];         //Merchant custom data
}paypass_trans_param;
#pragma pack()
```

15.1.1. paypass_trans_param_set

Prototype	int paypass_trans_param_set(const paypass_trans_param * param)
Function	Set transaction parameter.
Parameter	Param: Transaction parameters
Return Value	EMV_CL_OK_APPROVED EMV_CL_DECLINE EMV_CL_NEED_ONLINE EMV_CL_SELECT_NEXT EMV_CL_OTHER_INTERFACE EMV_CL_OTHER_CARD EMV_CL_TRY_AGAIN EMV_CL_NEED_ANOTHER_TAP EMV_CL_TERMINATED EMV_CL_ERROR EMV_CL_NO_SUPPORTED

	EMV_CL_CANCELED

15.1.2. paypass_trans_param_get

Prototype	int paypass_trans_param_get(paypass_trans_param * param)
Function	Get the trans parameter from kernel
Parameter	Param: Transaction parameters
Return Value	EMV_CL_OK_APPROVED EMV_CL_DECLINE EMV_CL_NEED_ONLINE EMV_CL_SELECT_NEXT EMV_CL_OTHER_INTERFACE EMV_CL_OTHER_CARD EMV_CL_TRY_AGAIN EMV_CL_NEED_ANOTHER_TAP EMV_CL_TERMINATED EMV_CL_ERROR EMV_CL_NO_SUPPORTED EMV_CL_CANCELED

15.1.3. paypass_trans_param_get_trans_type

Prototype	EMV_CL_TRANS_TYPE paypass_trans_param_get_trans_type(void)
Function	Return the transaction type
Parameter	NONE
Return Value	EMV_CL_PURCHASE EMV_CL_PURCHASE_WITH_CASHBACK EMV_CL_CASH_WITHDRAWVAL EMV_CL_CASH_DISBURSEMENT EMV_CL_REFUND

15.1.4. paypass_transaction_pre_transaction

Prototype	int paypass_transaction_pre_transaction(void);
Function	A dummy function, no used.

Parameter	NONE
Return Value	EMV_CL_OK_APPROVED EMV_CL_DECLINE EMV_CL_NEED_ONLINE EMV_CL_SELECT_NEXT EMV_CL_OTHER_INTERFACE EMV_CL_OTHER_CARD EMV_CL_TRY_AGAIN EMV_CL_NEED_ANOTHER_TAP EMV_CL_TERMINATED EMV_CL_ERROR EMV_CL_NO_SUPPORTED EMV_CL_CANCELED

15.1.5. paypass_entry_point

Prototype	int paypass_entry_point(int ifd, int select_next, unsigned char * aid, unsigned int * aid_len, unsigned char * FCI, unsigned int * FCI_Len);
Function	Select PPSE, Select the first application.
Parameter	Ifd : handle if the contactless reader Select_next: 0: No 1: Yes Aid : application aid Aid_len : length of the aid. FCI : FCI FCI_Len : Length of the FCI
Return Value	EMV_CL_OK_APPROVED EMV_CL_DECLINE EMV_CL_NEED_ONLINE EMV_CL_SELECT_NEXT EMV_CL_OTHER_INTERFACE EMV_CL_OTHER_CARD EMV_CL_TRY_AGAIN EMV_CL_NEED_ANOTHER_TAP EMV_CL_TERMINATED EMV_CL_ERROR EMV_CL_NO_SUPPORTED EMV_CL_CANCELED

15.1.6. paypass_transaction

Prototype	int paypass_transaction(int ifd, unsigned char * aid, unsigned int aid_len, const unsigned char * FCI, unsigned int FCI_Len);
Function	Card transaction
Parameter	Ifd : handle if the contactless reader Aid : application aid Aid_len: length of the aid. FCI : FCI FCI_Len: Length of the FCI
Return Value	EMV_CL_OK_APPROVED EMV_CL_DECLINE EMV_CL_NEED_ONLINE EMV_CL_SELECT_NEXT EMV_CL_OTHER_INTERFACE EMV_CL_OTHER_CARD EMV_CL_TRY_AGAIN EMV_CL_NEED_ANOTHER_TAP EMV_CL_TERMINATED EMV_CL_ERROR EMV_CL_NO_SUPPORTED EMV_CL_CANCELED

15.1.7. paypass_tranaction_complete

Prototype	int paypass_tranaction_complete(void);
Function	Complete the Paypass transaction after paypass_transaction
Parameter	NONE
Return Value	EMV_CL_OK_APPROVED EMV_CL_DECLINE EMV_CL_NEED_ONLINE EMV_CL_SELECT_NEXT EMV_CL_OTHER_INTERFACE EMV_CL_OTHER_CARD EMV_CL_TRY_AGAIN

	EMV_CL_NEED_ANOTHER_TAP EMV_CL_TERMINATED EMV_CL_ERROR EMV_CL_NO_SUPPORTED EMV_CL_CANCELED

15.1.8. paypass_app_type_get

Prototype	EMV_CL_APP_TYPE paypass_app_type_get(void);
Function	Get the application type
Parameter	NONE
Return Value	EMV_CL_APP_UNKNOW, EMV_CL_APP_MAGSTRIPE, EMV_CL_APP_EMV

15.1.9. paypass_clean

Prototype	int paypass_clean(unsigned char * CleanData, unsigned int * size, int * IsLast);
Function	CLEAN signal, remove the expired torn record from the kernel, and generate message data.
Parameter	CleanData: Message signal data, TLV format. Size : Length of the data IsLast : 0:Not last 1: Last One
Return Value	

15.1.10. paypass_torn_record_delete_all

Prototype	int paypass_torn_record_delete_all(void);
Function	Delete all torn record from the kernel buffer.
Parameter	NONE
Return Value	EMV_CL_OK_APPROVED EMV_CL_DECLINE EMV_CL_NEED_ONLINE EMV_CL_SELECT_NEXT

	EMV_CL_OTHER_INTERFACE EMV_CL_OTHER_CARD EMV_CL_TRY_AGAIN EMV_CL_NEED_ANOTHER_TAP EMV_CL_TERMINATED EMV_CL_ERROR EMV_CL_NO_SUPPORTED EMV_CL_CANCELED

15.1.11. paypass_msg_signal_data_get

Prototype	int paypass_msg_signal_data_get(unsigned char * buff, unsigned int * size);
Function	Get message signal data
Parameter	Buff: TLV format. Size: length of the TLV
Return Value	EMV_CL_OK_APPROVED EMV_CL_DECLINE EMV_CL_NEED_ONLINE EMV_CL_SELECT_NEXT EMV_CL_OTHER_INTERFACE EMV_CL_OTHER_CARD EMV_CL_TRY_AGAIN EMV_CL_NEED_ANOTHER_TAP EMV_CL_TERMINATED EMV_CL_ERROR EMV_CL_NO_SUPPORTED EMV_CL_CANCELED

15.1.12. paypass_final_trans_data_get

Prototype	int paypass_final_trans_data_get(unsigned char * buff, unsigned int * size);
Function	Generate final transaction data
Parameter	Buff: TLV format. Size: length of the TLV

Return Value	EMV_CL_OK_APPROVED EMV_CL_DECLINE EMV_CL_NEED_ONLINE EMV_CL_SELECT_NEXT EMV_CL_OTHER_INTERFACE EMV_CL_OTHER_CARD EMV_CL_TRY_AGAIN EMV_CL_NEED_ANOTHER_TAP EMV_CL_TERMINATED EMV_CL_ERROR EMV_CL_NO_SUPPORTED EMV_CL_CANCELED

15.1.13. paypass_ms_error_wait_start

Prototype	void paypass_ms_error_wait_start(void);
Function	Set timer for the error wait.
Parameter	NONE
Return Value	NONE

15.1.14. paypass_ms_error_wait

Prototype	void paypass_ms_error_wait(void);
Function	Called after closed NFC
Parameter	NONE
Return Value	NONE