User Guide of EMV Core API Library

Version 1.1

2012-05-03

Bluebird Soft Inc.

Table of Contents

1. History 4
2. Overview 5
3. Basic Knowledge 5
3.1 Header Files 5
3.2 Library Files 5
3.3 Dll Files 5
3.4 The Return Values of Function 5
4. Software Architecture 6
5. Application Programming Interface 6
5.1 Initialization Functions 6
5.1.1 Initialize EMV Library 6
5.1.2 Clear EMV Transaction Log 8
5.2 Parameters Related Functions 8
5.2.2 Set Terminal Parameters 9
5.2.3 Get TLV Parameters 9
5.2.4 Set TLV Parameters 9
5.2.5 Get Issuer Script Result 10
5.3 CAPK Application Functions 10
5.3.1 Add or Update CAPK 10
5.3.2 Delete CAPK 11
5.3.3 Get CAPK 11
5.3.4 Check CAPK 11
5.4 Application List Functions 12
5.4.1 Add or Update Application List 12
5.4.2 Delete Application List 13
5.4.3 Get Application List 13
5.5 EMV Core Transaction Functions 13
5.5.1 Select Application In The Application List 13
5.5.2 Read Application Data 13
5.5.3 Card Data Authentication 14
5.5.4 Processing Restrictions 14
5.5.5 Cardholder Verification 14
5.5.6 Terminal Risk Management 15
5.5.7 Process Transaction 15

5.6 EMV Core Interactional Callback Functions 15
5.6.1 GetDate And Time Of The Terminal 15
5.6.2 Special TLV Processing 16
5.6.3 Part of Terminal Parameters IO Controlling 16
5.6.4 User Input Transaction Amount 17
5.6.5 User Select The Application In The List 17
5.6.6 Transaction Advice Processing 17
5.6.7 User Input Card Holder PIN 18
5.6.8 Application Reference Processing 18
5.6.9 Transaction Online Processing 18
5.6.10 Display LCD (it will be explained next version) 19
5.6.11 Before GPO 19
5.6.12 Before 1st GenAC 20
5.6.13 Before 2nd GenAC 20
5.6.14 Plaintext PIN Verification 20
5.6.15 Enciphered PIN Verification 20
5.6.16 Reset IC Card Module 21
5.6.17 Close IC Card Module 21
5.6.18 Exchange IC Card Module Command 21
5.7 PIN Encryption Device Related Functions 22
5.7.1 Get Random Number 23
5.7.2 Write MK 23
5.7.3 Write PIN Key 23
5.7.4 Write MAC Key 24
5.7.5 Get PIN Block By PIN Key 24
5.7.6 Get Encrypted MAC By MAC Key 25
5.7.7 Offline Enciphered PIN 26
5.7.8 Offline Plaintext PIN 27
5.7.9 Get PIN Block By DUKPT 27
5.7.10 Get Encrypted MAC by DUKPT 28
5.7.11 Write DUKPT Key 29

1. History

Seq	Date	Version	Changes	Remarks
1	2012.01.09	V1.0	The first draft	
2	2012.05.03	V1.1	PCI related functions were appended.	

2. Overview

This document describes details of BBEMV API Function for terminal BIP-1500, including how to use terminal and BBEMV parameters access interface、CAPK management interface、EMV APP-List management interface、BBEMV interactional callback function interface and EMV APP processing interface(Such as creating APP list、reading APP data、card data authentication and on-line processing etc.).

3. Basic Knowledge

3.1 Header Files

BBEMV.h： declare of all EMV API and necessary macro definition.

3.2 Library Files

BBEMV.lib： The BBEMV library needs this file to link.

This requires you to link this file for the DLL, and you **could not** use function as **LoadLibrary** or **GetProcAddress** that supported in *run-time dynamic linking* methods.

3.3 Dll Files

VPOS329DLL.dll： This DLL is the file for internal reference. User doesn't need to refer to this file. This file should be copy to the same path with application executable file in final use.

329_EMVCORE_Dll.dll： This DLL is lower-level EMV functions which are connected to BBEMV.dll. This DLL is the file for internal reference. User doesn't need to refer to this file. This file should be copy to the same path with application executable file in final use.

BBEMV.dll： The set of EMV related functions. This file should be copy to the same path with application executable file in final use.

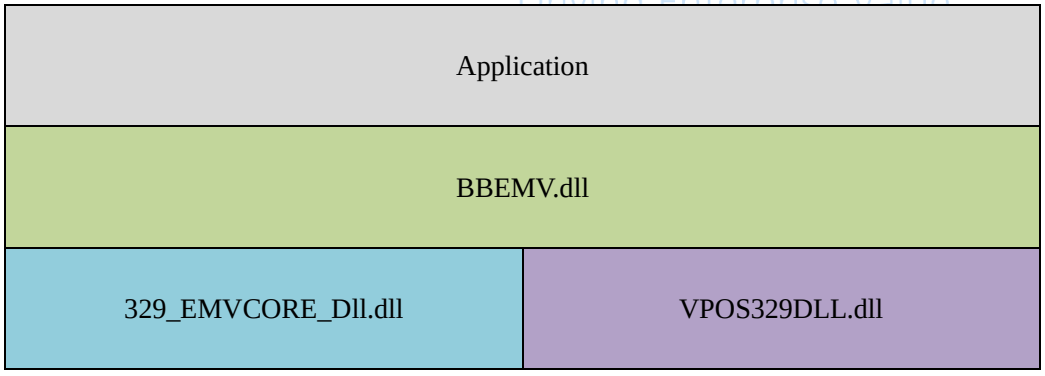
3.4 The Return Values of Function

```
#define EMV_OK              0       //success
#define ERR_EMVRSP          (-1)    //response error
#define ERR_APPBLOCK        (-2)    //application of card have blocked.
#define ERR_NOAPP           (-3)    //no application in card
#define ERR_USERCANCEL      (-4)    //operator or user cancel the transaction.
#define ERR_TIMEOUT         (-5)    //time out for inputting.
#define ERR_EMVDATA         (-6)    //data of card error
#define ERR_NOTACCEPT       (-7)    //the transaction is not accepted.
#define ERR_EMVDENIAL       (-8)    //the transaction is denial.
```

```
#define ERR_KEYEXP          (-9)    //the key is expired.
#define ERR_NOPINPAD        (-10)   //NO pinpad
#define ERR_NOPIN           (-11)   //by pass pin
#define ERR_CAPKCHECKSUM    (-12)   //the checksum of capk is error.
#define ERR_NOTFOUND        (-13)   //the Tag or data is not found.
#define ERR_NODATA          (-14)   //No data
#define ERR_OVERFLOW        (-15)   //the memory is overflowed.
#define ERR_NOTRANSLOG      (-16)   //no transaction log.
#define ERR_NORECORD        (-17)   //no transaction record.
#define ERR_NOLOGITEM       (-18)   //the item of log is error
#define ERR_ICCRESET        (-19)   //IC card reset fail
#define ERR_ICCCMD          (-20)   //IC card APDU command exchanging fail.
#define ERR_ICCBLOCK        (-21)   //IC card has been blocked.
#define ERR_ICCNORECORD     (-22)   //IC card has no record.
```

4. Software Architecture

- User Application can access BBEMV.dll for EMV core functions and callback functions.
- User Application can access bbappapi.dll for device driver functions, for example, beep, time,…
- User Application does not need to access the other .dlls because they are for internal reference.



5. Application Programming Interface

5.1 Initialization Functions

5.1.1 Initialize EMV Library

Declare：	**int BBEMV_Init(FUNCPTR_PARAM *ptrFunc**, unsigned char UsingPinPad**);**
Function：	Initialize EMV Library（Initialize EMV related parameters）.

Parameter：	**ptrFunc[in]:** See structure FUNCPTR_PARAM declare the below table. **UsingPinPad:** UsingPinPad[in] 0 … Not used PCI PinPad. 1 … Use PCI PinPad.
Return：	(EMV_OK) success
Remark：	Call it once in function MAIN( ) before all other EMV related processing.

>> structure FUNCPTR_PARAM definition：

```
typedef struct _FUNCPTR_PARAM_
{
    P_GETDATETIME      pGetDateTime;
    P_GETUNKNOWTLV     pGetUnknowTLV;
    P_IOCTRL           pIoCtrl;
    P_INPUTAMT         pInputAmt;
    P_WAITAPPSEL       pWaitAppSel;
    P_ADVICEPROC       pAdviceProc;
    P_GETHOLDERPWD     pGetHolderPwd;
    P_REFERPROC        pReferProc;
    P_ONLINEPROC       pOnlineProc;
    P_ACCOUNTYPESEL    pAccountTypeSel;
    P_PRINTXYCE        pLcdPrintxyCE;
    P_BEFOREGPO        pBeforeGpo;
    P_BEFORE1STGAC     pBefore1stGac;
    P_BEFORE2NDGAC     pBefore2ndGac;
    P_PLAINPINVERYFY   pPlainPinVerify;
    P_ENCPINVERIFY     pEncPinVerify;
    P_READIFDSN        pReadIfdSN;
    P_ICCRESET         pIccReset;
    P_ICCCLOSE         pIccClose;
    P_ICCCOMMAND       pIccCommand;
}FUNCPTR_PARAM;
```

>> The members of the structure are a series of pointers that point to the interactional callback function. The map of pointers and functions is as follow:

pGetDateTime	points to function	cbEMV_GetDateTime	//see title 2.6.10
pGetUnknowTLV	points to function	cbEMV _GetUnknowTLV	//see title 2.6.8
pIoCtrl	points to function	cbEMV _IoCtrl	//see title 2.6.11
pInputAmt	points to function	cbEMV _InputAmt	//see title 2.6.2
pWaitAppSel	points to function	cbEMV _WaitAppSel	//see title 2.6.1
pAdviceProc	points to function	cbEMV _AdviceProc	//see title 2.6.6
pGetHolderPwd	points to function	cbEMV _GetHolderPwd	//see title 2.6.3
pReferProc	points to function	cbEMV _ReferProc	//see title 2.6.4
pOnlineProc	points to function	cbEMV _OnlineProc	//see title 2.6.5
pAccountTypeSel	points to function	cbEMV _AccountTypeSel	//see title 2.6.12
pLcdPrintxyCE	points to function	cbEMV _LcdPrintxyCE	//see title 2.6.12
pBeforeGpo	points to function	cbEMV _BeforeGpo	//see title 2.6.12

pBefore1stGac	points to function	cbEMV _Before1stGac	//see title	2.6.12
pBefore2ndGac	points to function	cbEMV _Before2ndGac	//see title	2.6.12
pPlainPinVerify	points to function	cbEMV _PlainPinVerify	//see title	2.6.12
pPlainPinVerify	points to function	cbEMV _EncPinVerify	//see title	2.6.12
pReadIfdSN	points to function	cbEMV _ReadIfdSN	//see title	2.6.13
pIccReset	points to function	cbEMV _IccReset	//see title	2.6.14
pIccClose	points to function	cbEMV _IccClose	//see title	2.6.15
pIccCommand	points to function	cbEMV _IccCommand	//see title	2.6.16

For example)

1) Construct : FUNCPTR_PARAM m_tFuncPtr;
2) Make your Function : m_tFuncPtr.pGetDateTime = cbEMV_GetDateTime;
3) Code your Function in detail :

```
int cbEMV_GetDateTime(unsigned char *datetime)
{
        int nRet =    GetTime(datetime);
        return nRet;
}
```

4) Use your function : cbEMV_GetDateTime(datetime);

5.1.2 Clear EMV Transaction Log

Declare:	**void BBEMV_ClearTransLog (void);**
Function:	EMV library module will write log to the file to save some latest transaction PAN and amount, at step "terminal risk management" in transaction flow, to process the total amount and floor limit. Call this function to delete those logs.
Parameter:	None
Return:	None
Remark:	In practice, this function can be called after batch settlement.

5.2 Parameters Related Functions

5.2.1 Get Terminal Parameters

Declare:	**void BBEMV_GetParam(EMV_PARAM *tParam);**
Function:	Read terminal parameters stored in EMV library module.
Parameter:	**tParam [out]:** Pointer to data of EMV terminal parameter
Return:	None
Remark:	None

>> structure EMV_PARAM definition:

```
typedef struct{
    unsigned char MerchName[256];       //merchant name
```

```
    unsigned char MerchCateCode[2];    //Merchant code
    unsigned char MerchId[15];         //Merchant ID
    unsigned char TermId[8];           //Terminal ID
    unsigned char TerminalType;        //Terminal Type
    unsigned char Capability[3];       //Terminal Capability
    unsigned char ExCapability[5];     //Terminal Expent Capability
    unsigned char TransCurrExp;        //Transaction currency exponent
    unsigned char ReferCurrExp;        //Refer currency exponent
    unsigned char ReferCurrCode[2];    //Refer currency code
    unsigned char CountryCode[2];      //Terminal country code
    unsigned char TransCurrCode[2];    //Transaction currency code
    unsigned long ReferCurrCon;        //the conversion of refer currency.
    unsigned char TransType;           //current transaction type.
    unsigned char ForceOnline;         //Force online .
    unsigned char GetDataPIN;          //Before the PIN checking, it need read retry
times?
    unsigned char SurportPSESel;       //Support PSE selecting?
}EMV_PARAM;
```

5.2.2 Set Terminal Parameters

Declare:	**void BBEMV_SetParam(EMV_PARAM *tParam);**
Function:	Set terminal parameters stored in EMV library module.
Parameter:	**tParam[in]:** Pointer to data of EMV terminal parameter
Return:	None
Remark:	None

>> For structure EMV_PARAM definition, see the above 5.2.1.

5.2.3 Get TLV Parameters

Declare:	**int BBEMV_GetTLV(unsigned short Tag, unsigned char *DataOut, int *OutLen);**
Function:	Find an TLV element by parameter **Tag** in EMV library.
Parameter:	**Tag [in]:** Tag of the TLV to find. ***OutLen [out]:** Pointer to length of the TLV found. ***DataOut [out]:** Pointer to value of the TLV found.
Return:	(EMV_OK) Success (ERR_NODATA) Cannot find the TLV element.
Remark:	None

5.2.4 Set TLV Parameters

Declare:	**int BBEMV_SetTLV(unsigned short Tag, unsigned char**

	*DataIn, int DataLen);
Function:	Set and save a TLV elememt in EMV library.
Parameter:	**Tag [in]:** Tag of the TLV to save. **DataLen [in]:** Length of the TLV to save. ***DataIn [in]:** Pointer to value of the TLV to save.
Return:	(EMV_OK) Success (ERR_EMVDATA) Input data format error
Remark:	None

5.2.5 Get Issuer Script Result

Declare:	int BBEMV_GetScriptResult(unsigned char *Result, int *RetLen);
Function:	Get issuer script result.
Parameter:	***Result [out]:** Pointer to data of script result. ***RetLen [out]:** Pointer to length of script result.
Return:	(EMV_OK) success (ERR_ NODATA) There's no script result
Remark:	None

Format of the Result data of script result:

Byte 1 : Script Result

Most significant nibble: Result of the Issuer Script processing performed by the terminal:

'0' = Script not performed

'1' = Script processing failed

'2' = Script processing successful

Least significant nibble: Sequence number of the Script Command

'0' = Not specified

'1' to 'E' = Sequence number from 1 to 14

'F' = Sequence number of 15 or above

Bytes 2-5 : Script Identifier

Script Identifier of the Issuer Script received by the terminal, if available, zero filled if not. Mandatory if more than one Issuer Script was received by the terminal.

5.3 CAPK Application Functions

5.3.1 Add or Update CAPK

Declare:	int BBEMV_AddCapk(EMV_CAPK *capk);
Function:	Add or update CAPK data in EMV library.
Parameter:	***capk [in]:** Pointer to EMV_CAPK structure data.

Return：	(EMV_OK) success (ERR_CAPKCHECKSUM) check sum error (ERR_OVERFLOW) memory overflow
Remark：	EMV library can save at most 64 CAPK elements. if more than 64 CAPK elements be saved, it will return ERR_OVERFLOW.

>> structure EMV_CAPK definition：

```
typedef struct {
    unsigned char RID[5];           //application register ID
    unsigned char KeyID;            //key ID
    unsigned char HashInd;          //HASH arithmetic flag
    unsigned char ArithInd;         //RSA arithmetic flag
    unsigned char ModulLen;         //the length of Modul
    unsigned char Modul[248];       //the data of modul
    unsigned char ExponentLen;      //the length of exponet
    unsigned char Exponent[3];      //the data of exponent
    unsigned char ExpDate[3];       //Expire date(YYMMDD)
    unsigned char CheckSum[20];     //the check sum of key
}EMV_CAPK;
```

5.3.2 Delete CAPK

Declare：	**int BBEMV_DelCapk(unsigned char KeyID, unsigned char *RID);**
Function：	Delete one CAPK element by parameter **KeyID** and **RID** in EMV library.
Parameter：	**KeyID [in]：** key index of CAPK to delete. ***RID [in]：** Pointer to RID of the CAPK to delete.
Return：	(EMV_OK) success (ERR_NOTFOUND) Cannot find the CAPK to delete.
Remark：	None

5.3.3 Get CAPK

Declare：	**int BBEMV_GetCapk(int Index, EMV_CAPK *capk);**
Function：	Fine one CAPK element by parameter **Index** in EMV library.
Parameter：	**Index [in]：** key index of CAPK to find. ***capk [out]：** Pointer to the CAPK data found.
Return：	(EMV_OK) success (ERR_NOTFOUND) Cannot find the CAPK.
Remark：	None

5.3.4 Check CAPK

Declare：	**int BBEMV_CheckCapk(unsigned char *KeyID, unsigned**

	char *RID);
Function：	Check if specified CAPK element in EMV library is valid.
Parameter：	***KeyID [in]：** key index of CAPK to check. ***RID [out]：** RID of CAPK to check.
Return：	(EMV_OK) CAPK is valid. (ERR_KEYEXP) CAPK is invalid.
Remark：	None

5.4 Application List Functions

5.4.1 Add or Update Application List

Declare：	**int BBEMV_AddApp(EMV_APPLIST *App);**
Function：	Add or update application list.
Parameter：	***App [in]：** Pointer to the application list item to add.
Return：	(EMV_OK) success (ERR_OVERFLOW) memory overflow
Remark：	EMV library can save at most 32 application list items. if more than 32 items be saved, it will return ERR_OVERFLOW.

>> structure EMV_APPLIST definition：

```
typedef struct{
    unsigned char AppName[33];      //application name，end with '\x00'
    unsigned char AID[17];          //AID
    unsigned char AidLen;           //AID length
    unsigned char SelFlag;          //ASI
    unsigned char Priority;         //Priority
    unsigned char TargetPer;        //the Target percent
    unsigned char MaxTargetPer;     //the Max target percent
    unsigned char FloorLimitCheck;  //need  Floor limit amount checking?
    unsigned char RandTransSel;     //need random transaction selecting?
    unsigned char VelocityCheck;    //need velocity checking?
    unsigned long FloorLimit;       //the amount of floor limited.
    unsigned long Threshold;        //the value of threshold
    unsigned char TACDenial[6];     //TAC of terminal (denial)
    unsigned char TACOnline[6];     //TAC of terminal(online)
    unsigned char TACDefault[6];    //TAC of terminal (default)
    unsigned char AcquierId[7];     //Acquier ID
    unsigned char dDOL[256];        // DDOL of terminal ( dDOL format =
                                       Length(1byte)+data(nbyte) )
    unsigned char tDOL[256];        // TDOL of terminal ( tDOL format =
                                       Length(1byte)+data(nbyte) )
    unsigned char Version[3];       //application version
    unsigned char RiskManData[10];  //the data for risk management.( RiskManData
                                       format = Length(1byte)+data(nbyte) )
```

```
}EMV_APPLIST;
```

5.4.2 Delete Application List

Declare:	**int BBEMV_DelApp(unsigned char *AID, int AidLen);**
Function:	Delete one application list item.
Parameter:	***AID [in]:** Pointer to AID of App-list item to delete. **AidLen [in]:** Length of AID
Return:	(EMV_OK) success (ERR_NOTFOUND) cannot find the application list item
Remark:	None

5.4.3 Get Application List

Declare:	**int BBEMV_GetApp(int Index, EMV_APPLIST *App);**
Function:	Get application list item by parameter **Index**.
Parameter:	**Index [in]:** Index of the application list item in files. ***App [out]:** Pointer to application list item found.
Return:	(EMV_OK) success (ERR_NOTFOUND) cannot find the application list item
Remark:	None

5.5 EMV Core Transaction Functions

5.5.1 Select Application In The Application List

Declare:	**int BBEMV_AppSel(int Slot,unsigned long TransNo);**
Function:	Select application.
Parameter:	**Slot [in]:** User IC card slot num. **TransNo[in]:** The number of the transaction.
Return:	(EMV_OK) Select application successfully. (ERR_ICCRESET) Card reset failed. (ERR_ICCCMD) Card change command failed (ERR_ICCBLOCK) Card is locked. (ERR_NOAPP) There is no application supported in the terminal. (ERR_APPBLOCK) Application is locked. (ERR_EMVDATA) Card EMV data format error. (ERR_TIMEOUT) Select application selection timeout. (ERR_USERCANCEL) User cancel the transaction
Remark:	After card is detected in the slot, the function should be called to select application.

5.5.2 Read Application Data

Declare:	**int BBEMV_ReadAppData(void);**
Function:	Get the data of the application which are selected by function **BBEMV_AppSel**
Parameter:	None
Return:	(EMV_OK) Read application data successfully. (ERR_ICCCMD) Card change command failed. (ERR_EMVRSP) Card response code indicates error. (ERR_EMVDATA) Card data format error. (ERR_TIMEOUT) Input amount timeout. (ERR_USERCANCEL)User cancel the transaction when input amount
Remark:	If this function returns error code, the whole transaction flow should be terminated.

5.5.3 Card Data Authentication

Declare:	**int BBEMV_CardAuth(void);**
Function:	Card data authentication.
Parameter:	None
Return:	(EMV_OK) Card data authenticated successfully. (ERR_ICCCMD) Card change command failed. (ERR_EMVRSP) Card response code indicates error. (ERR_EMVDENIAL) Transaction is declined.
Remark:	None

5.5.4 Processing Restrictions

Declare:	**Void BBEMV_ProcRestriction(void);**
Function:	Processing restrictions..
Parameter:	None
Return:	None
Remark:	This function must be called after card authentication.

5.5.5 Cardholder Verification

Declare:	**int BBEMV_CardholderVerify(void);**
Function:	Perform cardholder verification
Parameter:	None
Return:	(EMV_OK) Cardholder verification perform successfully. (ERR_ICCCMD) Card command failed. (ERR_EMVRSP) Card response status code indicates error. (ERR_EMVDATA) Card data format error. (ERR_TIMEOUT) Input PIN timeout. (ERR_USERCANCEL)User cancel the transaction when inputing PIN

Remark:	If this function returns error code, the whole transaction flow should be terminated.

5.5.6 Terminal Risk Management

Declare:	**int BBEMV_TermRiskManage(void);**
Function:	Terminal risk management
Parameter:	None
Return:	(EMV_OK) Terminal risk management perform successfully. (ERR_ICCCMD) Card command failed. (ERR_EMVRSP) Card response status code indicates error. (ERR_EMVDATA) Card data format error.
Remark:	If this function returns error code, the whole transaction flow should be terminated.

5.5.7 Process Transaction

Declare:	**int BBEMV_ProcTrans(void);**
Function:	EMV transaction processing including terminal action analysis, card action analysis, online processing, script processing.
Parameter:	None
Return:	(EMV_OK) Transaction processed successfully. (ERR_ICCCMD) Card change command failed. (ERR_EMVRSP) Card response code indicates error. (ERR_EMVDATA) Card data format error. (ERR_NOTACCEPT) Transaction is not accepted. (ERR_EMVDENIAL) Transaction is declined. (ERR_TIMEOUT) Input cardholder PIN timeout. (ERR_USERCANCEL) User cancel the transaction.
Remark:	None

5.6 EMV Core Interactional Callback Functions

5.6.1 GetDate And Time Of The Terminal

Declare:	**int cbEMV_GetDateTime(unsigned char *datetime);**
Function:	When EMV library module check the valid date of card, it need call this function to know current date and time .

Parameter：	***datetime [out]：** Pointer to the date and time, BCD data format： datetime[0] → yeat(YY) (\x08=2008) datetime[1] → month(MM)(\x11=november) datetime[2] → date(DD) datetime[3] → hour(HH) datetime[4] → minute(MM) datetime[5] → second(SS) datetime[6] → days(WW) days-value map: 1 → monday, 2 → tuesday, …… 7 → sunday.
Return：	(EMV_OK) success (other value) failed
Remark：	Days (datetime[6]) is useless in EMV library modual. Year value range：1950－2049；if datetime[0] > 50, the year will be 19YY, or it'll be 20YY.

5.6.2 Special TLV Processing

Declare：	**int cbEMV_GetUnknowTLV(unsigned short Tag, unsigned char *dat, int len);**
Function：	When EMV library module cannot recognize a tag in DOL processing, it will call this function to query the special TLV item data. If application cannot recognize the tag either, it can return a non-zero value directly.
Parameter：	**Tag [in]：** tag of special TLV item. **Len [out]：** length of special TLV item. ***dat [out]：** value of special TLV item.
Return：	(EMV_OK) success (OTHER VALUE) There's no specified special TLV data.
Remark：	None

5.6.3 Part of Terminal Parameters IO Controlling

Declare：	**void cbEMV_IoCtrl(unsigned char ioname,unsigned char *iovalue);**
Function：	EMV library module call this function to get partly terminal parameters, such as POS input mode setting, advice support setting and batch data capture support setting, etc.
Parameter：	#define EMV_GET_POSENTRYMODE 0 #define EMV_GET_BATCHCAPTUREINFO 1 #define EMV_GET_ADVICESUPPORTINFO 2 **ioname [in]:** Name of control parameter ***iovalue [out]：** Pointer to data of specified parameter
Return：	None

Remark:	Application should support at least three parameters to query.

5.6.4 User Input Transaction Amount

Declare:	**int cbEMV_InputAmt(unsigned long *AuthAmt, unsigned long *CashBackAmt);**
Function:	User input transaction amount.
Parameter:	***AuthAmt [out]:** Pointer to transaction amount. ***CashBackAmt ([in]/[out]):** Pointer to cash back amount. If the terminal doesn't support cash back transaction, it can be input as a NULL pointer.
Return:	(EMV_OK) Input successfully (ERR_TIMEOUT) Input amount timeout (ERR_USERCANCEL) User cancel the transaction
Remark:	None

5.6.5 User Select The Application In The List

Declare:	**int cbEMV_WaitAppSel(int TryCnt, EMV_APPLIST List[], int AppNum);**
Function:	If there is one or more applications need user to confirm or select, EMV library module will call it and wait user selecting.
Parameter:	**TryCnt [in]:** = 0: Library call it for the first time !=0: Library call it after the first time (EMV specifications require that if the terminal call it more than one time in a transaction, it should prompt information "APP NOT ACCEPT"、" TRY AGAIN", etc.) **List[] [in]:** The candidate application list data **AppNum [in]:** Number of candidate application list items
Return:	(>=0) Index of the item selected in the list (ERR_TIMEOUT) Select application list timeout. (ERR_USERCANCEL) User cancel the transaction
Remark:	None

5.6.6 Transaction Advice Processing

Declare:	**void cbEMV_AdviceProc(void);**
Function:	Advice processing
Parameter:	None
Return:	None
Remark:	If the terminal doesn't support advice processing, the function should return directly, or it should save the advice log and decide to process whether batch up or online capture.

5.6.7 User Input Card Holder PIN

Declare:	**int cbEMV_GetHolderPwd(int TryFlag, int RemainCnt, unsigned char *pin);**
Function:	The terminal wait user input cardholder's online PIN.
Parameter:	**TryFlag [in]:** =0 EMV library modual call this function for the first time in one transaction. =1 EMV library modual call this function more than one time in one transaction.（Last PIN input is verified failed） **RemainCnt [in]:** PIN retry counter. If the value is 1, User can input PIN for the last time, if the PIN is veritied failed again, the card will be locked. ***pin ([in]/[out]): NULL** The input pointer value is NULL when transaction need a online PIN, then parameter **TryFlag** and **RemainCnt** is useless. The online PIN should be saved and encrypted in application, it will be checked by the EMV host when the transaction being process online.
Return:	(EMV_OK) PIN is input successfully (ERR_NOPINPAD) There's no pinpad or pinpad is broken. (ERR_NOPIN) There's no PIN or BYPASS (ERR_TIMEOUT) User input pin timeout (ERR_USERCANCEL) User cancel the transaction
Remark:	None

5.6.8 Application Reference Processing

Declare:	**int cbEMV_ReferProc(void) ;**
Function:	Process reference sent by issuer. none
Parameter:	None
Return:	(REFER_APPROVE) Transaction approved (REFER_DENIAL) Transaction declined
Remark:	If the terminal doesn't support to process the reference, it will return directly.

5.6.9 Transaction Online Processing

Declare:	**int cbEMV_OnlineProc(unsigned char *RspCode, unsigned char *AuthCode, int *AuthCodeLen, unsigned char *IAuthData, int *IAuthDataLen, unsigned char *Script, int *ScriptLen);**

Function：	Transaction online processing.
Parameter：	***RspCode [out]：** Pointer to Authentication response code (2Bytes), if there is no ARC, RspCode[0] is 0 (for example, transaction online processed failed). ***AuthCode [out]：** Pointer to authentication code（6Bytes） ***AuthCodeLen [out]：** Pointer to length of authentication code, if there is no authentication code, AuthCodeLen is 0. ***IAuthData [out]：** Pointer to external authenticate data return form EMV host. ***IAuthDataLen [out]：** Pointer to length of external authenticate data return form EMV host. The value is 0 if there is no external authenticate data. ***Script [out]：** Pointer to issuer script. If the whole script return by more than one 8583 packet, application shold jion the pack data to one whole script and output it to this pointer. ***ScriptLen [out]：** Pointer to length of issuer scrip, if there is no issuer scrip, the value is 0.
Return：	(ONLINE_APPROVE) Online approved（EMV host approved it） (ONLINE_DENIAL) Online declined（EMV host declined it） (ONLINE_REFER) Online referral（issuer referral） (ONLINE_FAILED) Online failed (ONLINE_ABORT) Online transaction abort
Remark：	In this function, application should call function EmvLib_GetTLV() to get necessary IC card data. Then make packet supported by the EMV host and send ,receive the packet, and output the related data to the EMV library module.

5.6.10 Display LCD (it will be explained next version)

Declare：	**int cbEMV_LcdPrintxyCE(unsigned char col, unsigned char row, unsigned char mode, char *pCHN, char *pEN);**
Function：	
Parameter：	
Return：	

5.6.11 Before GPO

Declare：	**void cbEMV_BeforeGPO(PUBLIC_KEY * pk)**
Function：	This function is called by kernel before GPO command. Application may get TLV data by implementing this function. If nothing to be done, application may just return in this function.

Parameter:	None
Return:	None

5.6.12 Before 1st GenAC

Declare:	int cbEMV_Before1stGenAC(void)
Function:	This function is called by kernel before 1st GenAC command. Application may get TLV data by implementing this function. If nothing to be done, application may just return in this function.
Parameter:	None
Return:	None

5.6.13 Before 2nd GenAC

Declare:	int cbEMV_Before2ndGenAC(void)
Function:	This function is called by kernel before 2nd GenAC command. Application may get TLV data by implementing this function. If nothing to be done, application may just return in this function.
Parameter:	None
Return:	None

5.6.14 Plaintext PIN Verification

Declare:	int cbEMV_PlainPinVerify(void)
Function:	Offline plaintext PIN verification. This function get PIN from PED and send PIN to IC Card for verification offline.
Parameter:	None
Return:	(EMV_OK) PIN verification successfully. (ERR_ICCCMD) Card command failed. (ERR_EMVRSP) Card response status code indicates error. (ERR_EMVDATA) Card data format error. (ERR_TIMEOUT) Input PIN timeout. (ERR_USERCANCEL)User cancel the transaction when input PIN
Remark:	If this function returns error code, the whole transaction flow should be terminated.

5.6.15 Enciphered PIN Verification

Declare:	int cbEMV_EncipherPinVerify(PUBLIC_KEY * pk)
Function:	Offline enciphered PIN verification. This function gets PIN from PED and enciphers the PIN using public key, and then sends enciphered PIN to IC Card for verification offline.

Parameter:	None
	Pk: RSA public key used for encipher PIN.
	Typedef struct{
	unsigned char ModulLen; //Module length
	unsigned char Modul[248]; //Module
	unsigned char ExponentLen; //Exponent length
	unsigned char Exponent[3]; //Exponent
	} PUBLIC_KEY

Return:	(EMV_OK) PIN verification successfully. (ERR_ICCCMD) Card command failed. (ERR_EMVRSP) Card response status code indicates error. (ERR_EMVDATA) Card data format error. (ERR_TIMEOUT) Input PIN timeout. (ERR_USERCANCEL)User cancel the transaction when input PIN
Remark:	If this function returns error code, the whole transaction flow should be terminated.

5.6.16 Reset IC Card Module

Declare:	**int cbEMV_IccReset(unsigned char slot,unsigned char VCC_Mode,unsigned char *ATR)**
Function:	Ic card reader should be reset in this function.
Parameter:	**Slot[in]** slot – Card slot number.
	Vcc_mode[in] Card power supply voltage: 1---5V 2---3V 3---1.8V
	ATR[out] Pointer to card reset response.（Need a 32 +1 bytes buffer） It contains length(1byte)+ response data
Return:	0 Success Other error

5.6.17 Close IC Card Module

Declare:	**int cbEMV_IccClose(unsigned char slot)**
Function:	IC Card module should be closed using this function.
Parameter:	Slot[in] Card slot number
Return:	None

5.6.18 Exchange IC Card Module Command

Declare:	**int cbEMV_IccCommand(unsigned char slot,ICC_APDU_SEND * ApduSend,ICC_APDU_RESP ***

	ApduResp);
Function：	IC card operate. The function support IC card general interface protocol. (T=0 and T=1)
Parameter：	Slot[in] Card slot number **ApduSend[out]** see Remark table **ApduResp[out]** see Remark table
Return：	0-success other-fail

Remark：	>>APDU_SEND structure：
	struct{
	BYTE Command[4];
	WORD Lc;
	BYTE DataIn[512];
	WORD Le;
	};
	Command[] = {CLA，INS，P1，P2}
	Lc = length of DataIn
	DataIn = data sent to the card
	Le = wanted length of received data.
	Case1：Lc=0; Le=0
	No send data and no receive data.
	Case2: Lc=0; Le>0
	No send data but wanted receive data. If the length of receive data is unknown,then Le=256.
	Case3: Lc>0; Le=0
	Send some data and no received data.
	Case4: Lc>0; Le>0
	Send some data and wanted receive data.
	Note：
	Le=256 when no receive data wanted.
	
	>>APDU_RESP structure：
	struct{
	WORD LenOut;
	BYTE DataOut[512];
	BYTE SWA;
	BYTE SWB;
	};
	LenOut = The actual length response data from IC card
	DataOut = Pointer to response data from IC card.
	SWA = Status byte 1
	SWB = Status byte 2

5.7 PIN Encryption Device Related Functions

5.7.1 Get Random Number

Declare:	**void BBPCIGetRandom (unsigned char *rnd8);**
Function:	Get 8 bytes random
Parameter:	**rnd8[out] :** The buffer of random
Return:	None

5.7.2 Write MK

Declare:	**int BBPCIWrite_M_Key (unsigned char key_no, unsigned char key_len, unsigned char *key_data, unsigned char mode);**
Function:	Write Master Key
Parameter:	**key_no[in]:** Key index, range: 0~49 **key_len[in]:** Key length, only can be 8、16、24 **key_data[in]:** Key data **mode[in]:** Write mode 0：write main key directly 0xff: perform an exclusive-OR operation on **key_data** and old key and write the result.
Return:	0: Success -7003: Illegal key index -7004: Illegal key length -7000: PED LOCKED

5.7.3 Write PIN Key

Declare:	**int BBPCIWrite_PIN_Key (unsigned char key_no, unsigned char key_len, unsigned char *key_data, unsigned char mode, unsigned char mkey_no);**
Function:	Write PIN Key
Parameter:	**key_no[in]:** Key index, range: 0~49 **key_len[in]:** Key length, only can be 8、16、24 **key_data[in]:** Key data **mode[in]:** Write mode 0X00：write directly 0X01：encrypt by MK 0X81：decrypt by MK

Return：	0: Success -7002: CRC Error -7003: Illegal key index -7004: Illegal key length -7000: PED LOCKED
Comment	If the length of MK is 8 bytes, it can not used to encrypt or decrypt the PINK with lengh 16 bytes or 24 bytes.

5.7.4 Write MAC Key

Declare：	**int BBPCIWrite_MAC_Key (unsigned char key_no, unsigned char key_len, unsigned char *key_data, unsigned char mode, unsigned char mkey_no);**
Function：	Write MAC Key
Parameter：	**key_no[in]:** Key index, range: 0~49 **key_len[in]:** Key length, only can be 8、16、24 **key_data[in]:** Key data **mode[in]:** Write mode 0X00：write directly 0X01：encrypt by MK 0X81：decrypt by MK
Return：	0: Success -7002: CRC Error -7003: Illegal key index -7004: Illegal key length -7000: PED LOCKED
Comment	If the length of MK is 8 bytes, it can not used to encrypt or decrypt the MACK with lengh 16 bytes or 24 bytes..

5.7.5 Get PIN Block By PIN Key

Declare：	**int BBPCIGet_PIN (unsigned char pinkey_n, unsigned char min_len, unsigned char max_len, unsigned char *card_no, unsigned char mode, unsigned char *pin_block, unsigned short waittime_sec);**
Function：	Get PIN－BLOCK by using PIN Key。

Parameter：	**pinkey_n[in]:** PIN Key index, range: 0~49 **min_len[in]:** The minimum length of PIN，range: 1~14 **max_len[in]:** The maximum length of PIN，range: 1~14 **card_no[in]:** The card No shifted by POS，the first 4 bytes is 0000，and the last 12 bytes is the 12 bytes count backwards from the penultimate of card No（0000 PPPP PPPP PPPP） **mode[in]:** Encrypt mode(1bytes) 0=X9.8 1=X3.92(not support) mode's highest bit(bit7) indicate that allow/ interdict return by pressing "ENTER" if no pin input. Bit7=0 allow. Bit7=1 not allow **pin_block[out]:** PIN block result(8bytes) **waittime_sec[in]:** Time to wait.（Unit is second）If the input value is 0, it will be set to default 60s.(2bytes)
Return：	0: Success -7003: Illegal key index -7005: Illegal key mode -7006: Illegal input length -7007: Input cancel -7009: Input timeout -7000: PED LOCKED -7010: Input interval error -7016: Press Enter to quit without input PIN
Comment	this API can not be called over 120 times in 60 minutes

5.7.6 Get Encrypted MAC By MAC Key

Declare：	**int BBPCIGet_MAC (unsigned char mackey_n, unsigned short inlen, unsigned char *indata, unsigned char *macout, unsigned char mode);**
Function：	Encrypt mac message by using MAC Key
Parameter：	**mackey_n[in]:** The key index of MAC Key,range :0~49 **inlen[in]:** The length of mac message to be encrypted, must less than 2048 bytes. **indata[in]:** The mac message **macout[out]:** The encrypted result **mode[in]:** 0＝algorithm 1, 1＝algorithm 2, 2＝algorithm 3

Return：	0: Success -7003: Illegal key index -7005: Illegal encryption mode -7011: Key does not exist -7015: Illegal length -7000: PED LOCKED
Comment	✧ Divide the data into several BLOCK with 8 bytes,if the last BLOCK less than 8 bytes, fill remaining places with 0x00. Algorithm selection: 0：Algorithm 1； 1：Algorithm 2； 2：Algorithm 3。 ✧ Algorithm 1：encrypt the BLOCK1 with MAC encryption key using Algorithm DES or 3DES,XOR the result of encryption with BLOCK2, and encrypt the result of XOR with MAC encryption key using Algorithm DES or 3DES,repeat and get the last encrypt result of 8 bytes. ✧ Algorithm 2：XOR the BLOCK1 with BLOCK2，and XOR the result with BLOCK3，repeat，untill get last result of 8 bytes，encrypt the result with MAC encryption key using algorithm DES or algorithm 3DES. ✧ Algorithm 3：encrypt the BLOCK1 with MAC encryption key using algorithm DES,XOR the result with BLOCK2, then encrypt the result with MAC encryption key using algorithm DES,repeat,untill the last step,Get the encrypt result of 8 bytes using Algorithm TDES。

5.7.7 Offline Enciphered PIN

Declare：	**int BBPCIOffLine_Enc_PIN (unsigned char icc_slot, unsigned char min, unsigned char max, unsigned short waittime_sec, RSA_PINKEY *rsa_pinkey);**
Function：	Offline encrypted PIN
Parameter：	**Icc_slot[in]:** Icc slot:0 **min[in]:** The minimum length of PIN，range: 1~14 **max[in]:** The maximum length of PIN，range: 1~14 **waittime_sec[in]:** Time to wait.（Unit is second）If the input value is 0, it will be set to default 60s.(2bytes) **rsa_pinkey[in]:** The public key to encrypt the PIN

Return：	0: Success -7007: Input cancel -7009: Input timeout -7010: Input interval error -7016: Press Enter to quit without input PIN
Comment	

5.7.8 Offline Plaintext PIN

Declare：	**int BBPCIOffLine_Plain_PIN (unsigned char icc_slot, unsigned char min, unsigned char max, unsigned short waittime_sec);**
Function：	Offline Plaintext PIN
Parameter：	**Icc_slot[in]:** Icc slot:0 **min[in]:** The minimum length of PIN，range: 1~14 **max[in]:** The maximum length of PIN，range: 1~14 **waittime_sec[in]:** Time to wait.（Unit is second）If the input value is 0, it will be set to default 60s.(2bytes)
Return：	0: Success -7007: Input cancel -7009: Input timeout -7010: Input interval error -7016: Press Enter to quit without input PIN
Comment	

5.7.9 Get PIN Block By DUKPT

Declare：	**int BBPCIGet_PIN_DUKPT (unsigned char key_n, unsigned char min_len, unsigned char max_len, unsigned char *card_no, unsigned char mode, unsigned char *pin_block, unsigned short waittime_sec, unsigned char *out_ksn);**
Function：	Get PIN－BLOCK by using DUKPT

Parameter：	**key_n[in]:** The key index of DUKPT, range: 0~9 **min_len[in]:** The minimum length of PIN，range: 1~14 **max_len[in]:** The maximum length of PIN，range: 1~14 **card_no[in]:** The card No shifted by POS，the first 4 bytes is 0000，and the last 12 bytes is the 12 bytes count backwards from the penultimate of card No（0000 PPPP PPPP PPPP） **mode[in]:** Encrypt mode(1bytes) 0=X9.8 1=X3.92(not support) **pin_block[out]:** PIN block result(8bytes) **waittime_sec[in]:** Time to wait.（Unit is second）If the input value is 0, it will be set to default 60s.(2bytes) **out_ksn[out]:** ksn
Return：	0: Success -7003: Illegal key index -7005: Illegal key mode -7006: Illegal input length -7007: Input cancel -7009: Input timeout -7000: PED LOCKED -7010: Input interval error -7016: Press Enter to quit without input PIN
Comment	This API can not be called over 120 times in 60 minutes

5.7.10 Get Encrypted MAC by DUKPT

Declare：	**int BBPCIGet_MAC_DUKPT (unsigned char key_n, unsigned short inlen, unsigned char *indata, unsigned char *macout, unsigned char mode, unsigned char *out_ksn);**
Function：	Encrypt mac message by using DUKPT
Parameter：	**key_n[in]:** The key index of DUKPT, range: 0~9 **in_len[in]:** The length of mac message to be encrypted, must less than 2048 bytes. **indata[in]:** The mac message **macout[in]:** The encrypted result **mode[in]:** 0 = algorithm 1, 1 = algorithm 2, 2 = algorithm 3 **out_ksn[out]:** ksn

Return：	0: Success -7003: Illegal key index -7005: Illegal encryption mode -7000: PED LOCKED -7011: Key not exist -7015: Illegal length
Comment	✧ Divide the data into several BLOCK with 8 bytes,if the last BLOCK less than 8 bytes, fill remaining places with 0x00. Algorithm selection: 0：Algorithm 1； 1：Algorithm 2； 2：Algorithm 3。 ✧ Algorithm 1：encrypt the BLOCK1 with MAC encryption key using Algorithm DES or 3DES,XOR the result of encryption with BLOCK2, and encrypt the result of XOR with MAC encryption key using Algorithm DES or 3DES,repeat and get the last encrypt result of 8 bytes. ✧ Algorithm 2：XOR the BLOCK1 with BLOCK2，and XOR the result with BLOCK3，repeat，untill get last result of 8 bytes，encrypt the result with MAC encryption key using algorithm DES or algorithm 3DES. Algorithm 3：encrypt the BLOCK1 with MAC encryption key using algorithm DES,XOR the result with BLOCK2, then encrypt the result with MAC encryption key using algorithm DES,repeat,untill the last step,Get the encrypt result of 8 bytes using Algorithm TDES

5.7.11 Write DUKPT Key

Declare：	**int BBPCISecurity_Load_DUKPT_Key (unsigned char key_index, unsigned char *key, unsigned char key_len, unsigned char *ksn, unsigned char ksn_len);**
Function：	Write Dukpt Key
Parameter：	**Key_index[in]:** The key index of DUKPT, range: 0~9 **Key[in]:** The content of bdk **Key_len[in]:** The length of bdk **Ksn[in]:** The content of ksn **Ksn_len[in]:** The length of ksn
Return：	0: Success -7003: Illegal key index -7004: Illegal input length
Comment	