# PAX EMV Kernel API Programming Guide

**V 6.0.2**

**PAX Computer Technology（Shenzhen）Co., Ltd.**

# Revision History

| Date | Version | Note | Author |
|---|---|---|---|
| 2011-07-06 | V4.1.2 | 1. Resolve a problem that Scripts are processed error when there are data 71 or 72 contained in script.<br>2. Add processing of NULL point for all functions with point parameters.<br>3. The version is updated to v412. | |
| 2011-07-25 | V4.1.2_T1 | Add support for Contactless PBOC, including:<br>1. Add CLSS function.<br>2. Add macro definitions CLSS_SLOT and CONTACT_SLOT.<br>3. Add CLSS transaction parameter structure Clss_TransParam.<br>4. Add call-back function cEMVPiccIsoCommand.<br>5. Add API interface EMVSwitchClss.<br>6. The version is updated to v412_T1. | |
| 2011-12-02 | V4.1.3 | 1. Add API EMVGetMCKParam.<br>2. Add API EMVInitTLVData for Clss PBOC.<br>3. The version is updated to v413 | |
| 2012-02-14 | V4.1.3_T1 | Change the internal value setting for Terminal Capability and Transaction. | |
| 2012-06-14 | V4.1.4 | Modify the range of random number. | |
| 2012-06-08 | V5.0.0 | Based on v414,<br>1. Update the kernel according to the latest EMV 4.3a Specification.<br>2. Update the generation algorithm of UN<br>3. Add API EMVSetAmount to allow large amount setting (larger than 0xFFFFFFFF). | |
| 2012-06-15 | V5.0.0_T1 | 1. Revise the large amount setting issues. | |
| 2012-08-30 | V5.0.1 | 1. Update the structure definition for ELEMENT_ATTR. Please take attention to the parameter setting of function EMVAddIccTag().<br>2. Add support for the PAX Prolin software platform, including:<br>Add call-back functions:<br>cEMVIccIsoCommand.<br>cEMVPedVerifyPlainPin.<br>cEMVPedVerifyCipherPin.<br>The following functions are not provided:<br>EMVSetDebug | |

| | | | |
|---|---|---|---|
| | | EMVProcTrans<br>cEMVOnlineProc<br>EMVCoreVersion<br>EMVGetICCStatus<br>EMVSetPCIModeParam<br>EMVClearTransLog<br>3. AID list, CA public key, EMV parameters are no longer stored by the kernel. The application must store all these parameters.<br>Please note the significant change for the usage of the function cEMVUnknowTLVData.<br>4. When the platform is prolin, it is necessary to add the head file "emvlib_prolin.h". | |
| 2013-5-7 | V5.0.3 | 1. Add SM and Loading log for PBOC 3.0 requirement.<br>These parts add APIs below:<br>EMVReadSingleLoadLog<br>EMVGetSingleLoadLogItem<br>EMVReadAllLoadLogs<br>cEMVSM2Verify.<br>cEMVSM3<br>2. EMVSetScriptProcMethod function is added to distinguish the Union pay special script process from EMV. This is for requirement of Union Pay network authentication.<br>3. Expand the number of terminal AIDs supported. | |
| 2013-06-21 | V5.0.3_T1 | 1. Add return value definition EMV_FILE_ERR<br>2. Increase the operation if the old terminal application file exists then converted it to the new format. Fix the bug of V503; V503 due to the format of kernel's terminal supported application file has been updated, it will lead to if not rebuild the file system, the terminal supported application file format error occurs and cannot be loaded correctly. | |
| 2013-06-27 | V5.0.3_T2 | 1. Modify the description of EMVStartTrans and EMVCompleteTrans | |
| 2013-07-18 | V5.0.4 | 1. Modify the description of EMVCompleteTrans.<br>2. Add the description of the macro-definition of AC type.<br>3. Modify the description of FloorLimitCheck, RandTransSel and | |

| | | | |
|---|---|---|---|
| | | VelocityCheck of EMV_APPLIST. | |
| 2013-08-16 | V5.0.4_T1 | 1. Add API function EMVGetLogData to get log data.<br>2. Add API function EMVGetVerifyICCStatus to get the response status word SWA and SWB of PIN Verify command.<br>3. Delete the process of Terminal Action Analysis whether the transaction should be rejected immediately in EMVCardAuth. | |
| 2013-09-05 | V5.0.4_T2 | 1. Add API EMVDelAllApp to remove all applications from terminal application list.<br>2. Change the process of EMVDelApp to make sure that the order of the terminal application list is the same as the old version before v503 of EMV kernel. | |
| 2013-12-13 | V5.0.5 | 1. An online-only terminal need not support SDA or DDA or CDA. Individual payment systems will define rules for this case, the kernel will execute offline data authentication according to Terminal Capabilities. | |
| 2014-06-26 | V6.0.0 | 1. Add API EMVGetDebugInfo to get debug information for offline data authentication.<br>2. Modify the descriptions of EMV_REVOCLIST and EMVSetAmount.<br>3. Modify return codes or parameter of callback functions for appendix A.<br>Modify the value for REFER_APPROVE、REFER_DENIAL、ONLINE_APPROVE、ONLINE_FAILED、ONLINE_REFER、ONLINE_DENIAL and ONLINE_ABORT. | |
| 2015-03-20 | V6.0.2 | 1. Add EMVGetParamFlag to get the flag for signature and advice.<br>2. Modify the descriptions for function EMVSetConfigFlg’s parameter.<br>The default kernel supports advice.<br>3. Modify the descriptions for EMV_CAPK. | |

# Contents

# 1 Example of Application Programming for EMV

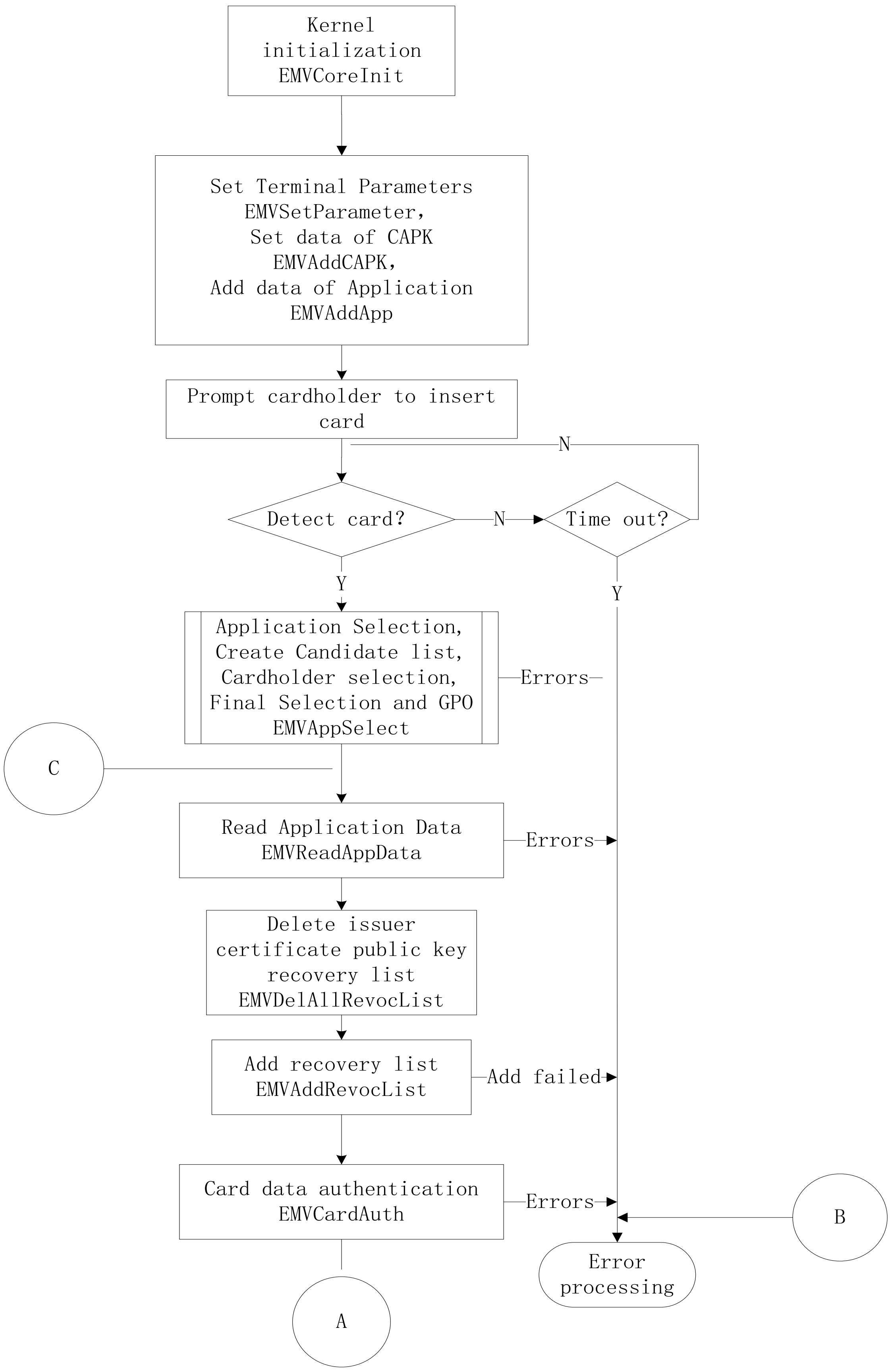
Kernel initialization EMVCoreInit
Set Terminal Parameters EMVSetParameter, Set data of CAPK EMVAddCAPK, Add data of Application EMVAddApp
Prompt cardholder to insert card
Detect card?
N
Time out?
N
Y
Y
Application Selection, Create Candidate list, Cardholder selection, Final Selection and GPO EMVAppSelect
Errors
C
Read Application Data EMVReadAppData
Errors
Delete issuer certificate public key recovery list EMVDelAllRevocList
Add recovery list EMVAddRevocList
Add failed
Card data authentication EMVCardAuth
Errors
B
Error processing
A

Figure1: EMV processing flow

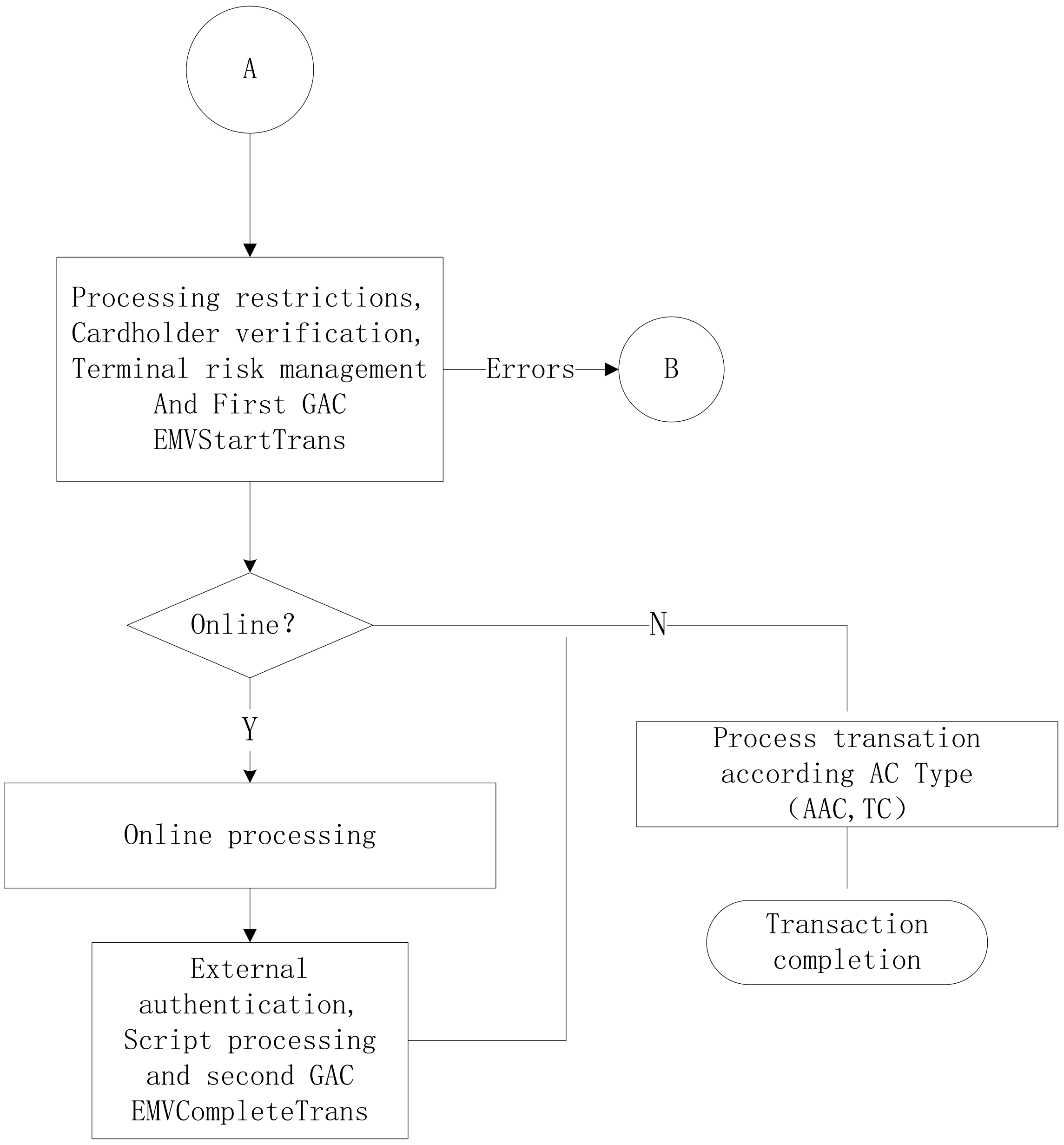
A
Processing restrictions, Cardholder verification, Terminal risk management And First GAC EMVStartTrans
Errors
B
Online?
N
Y
Online processing
External authentication, Script processing and second GAC EMVCompleteTrans
Process transation according AC Type (AAC, TC)
Transaction completion

Figure2: EMV processing flow2 ( for monitor and prolin platform )

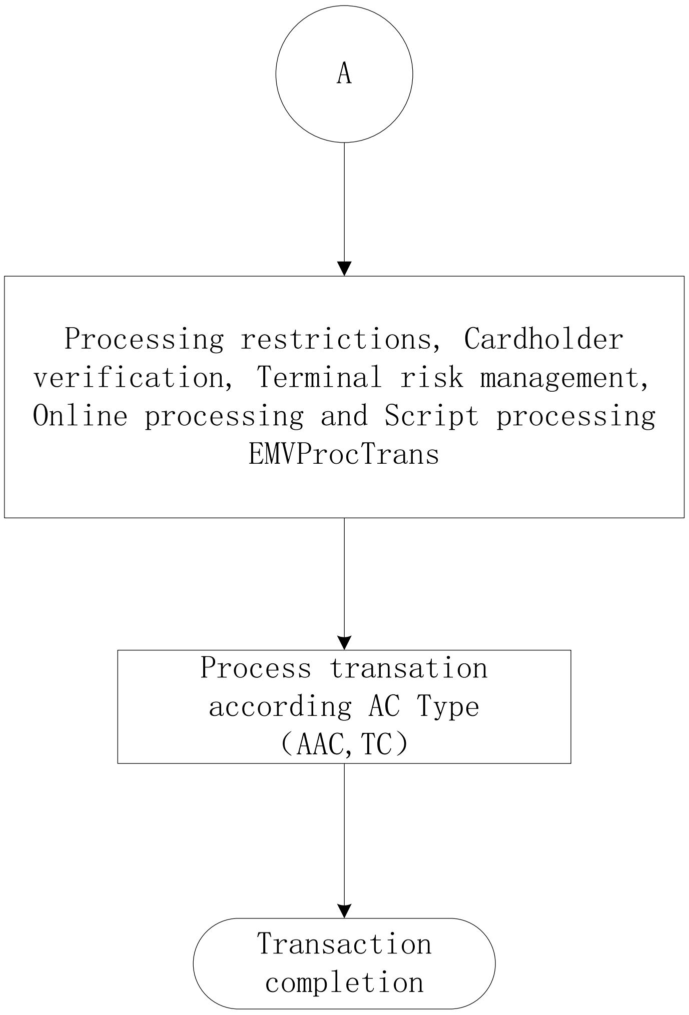
A
Processing restrictions, Cardholder verification, Terminal risk management, Online processing and Script processing EMVProcTrans
Process transation according AC Type (AAC,TC)
Transaction completion

Figure3: EMV processing flow2 ( only for monitor platform )

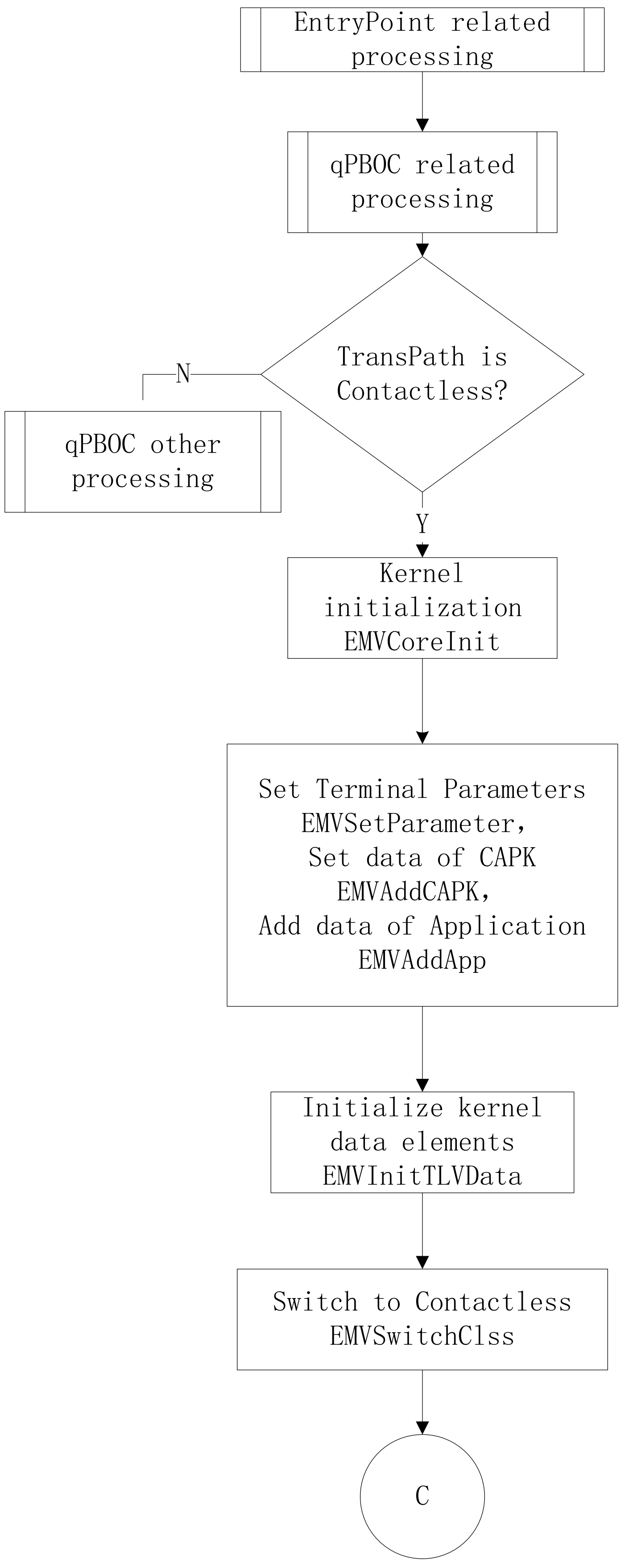
EntryPoint related processing
qPBOC related processing
TransPath is Contactless?
N
qPBOC other processing
Y
Kernel initialization EMVCoreInit
Set Terminal Parameters EMVSetParameter, Set data of CAPK EMVAddCAPK, Add data of Application EMVAddApp
Initialize kernel data elements EMVInitTLVData
Switch to Contactless EMVSwitchClss
C

Figure4: Contactless processing flow

# 2 Parameter and Data Management

Functions of parameter and data management of EMV kernel are described in this part.

## 2.1 EMVCoreInit

| | | |
|---|---|---|
| **Prototype** | **int EMVCoreInit(void);** | |
| **Function** | EMV kernel initialization. | |
| **Parameters** | None | |
| **Return** | EMV_OK | Initialization succeeds. |
| | EMV_KEY_EXP | There's expired key in the EMV kernel. |
| | EMV_FILE_ERR | File error |
| **Instruction** | 1. It initializes all the EMV data elements.<br>2. This function must be called once each time when the terminal powers on.<br>3. If there is expired key, the application can check it through function **EMVCheckCAPK** and use the key management functions to delete the expired key and/or add the new key.<br>4. If the return value is EMV_FILE_ERR, it means that there is problem when transfer the terminal supported application file from old format to new one. In this case terminal may need rebuild file system, and reset all parameters including CAPK, Applications and so on into terminal. | |

## 2.2 EMVCoreVersion

| | | |
|---|---|---|
| **Prototype** | **int EMVCoreVersion(void);** | |
| **Function** | Query the version of EMV kernel. | |
| **Parameters** | None | |
| **Return** | Version number. For example, 25 means version 2.5. | |
| **Instruction** | The function is Not Used for the Prolin software platform. | |

## 2.3 EMVGetParameter

| | | |
|---|---|---|
| **Prototype** | **void EMVGetParameter(EMV_PARAM *Param);** | |
| **Function** | Get terminal parameter. | |
| **Parameters** | Param[output] | The pointer pointing to terminal parameter. **Refer to appendix for structure EMV_PARAM.** |
| **Return** | None | |
| **Instruction** | | |

## 2.4 EMVSetParameter

| | | |
|---|---|---|
| **Prototype** | **void EMVSetParameter(EMV_PARAM *Param);** | |
| **Function** | Set terminal parameter. | |
| **Parameters** | Param[input] | The pointer pointing to terminal parameter. **Refer to appendix for structure EMV_PARAM.** |
| **Return** | None | |
| **Instruction** | | |

## 2.5 EMVGetTLVData

| | | |
|---|---|---|
| **Prototype** | **int EMVGetTLVData(unsigned short Tag, unsigned char *DataOut, int *DataLen);** | |
| **Function** | Get the value of the data element by specifying the tag. | |
| **Parameters** | Tag[input] | Tag of EMV standard or extended data element. |
| | DataOut[output] | The value of the data element specified by the tag. |
| | DataLen[output] | The length of the data element. |

| | | |
|---|---|---|
| **Return** | EMV_OK | Succeed. |
| | EMV_NO_DATA | Tag data not founded |
| | EMV_PARAM_ERR | Parameter error. |
| **Instruction** | EMV library can store all of the EMV standard data elements; it can also store up to 64 data elements which are defined by issuer. After executing the function of read application data, the application can call this function to get the value of the data element by specifying the tag.<br>**Refer to appendix A for the macro definition of the return values, same as below.** | |

## 2.6 EMVSetTLVData

| | | |
|---|---|---|
| **Prototype** | **int EMVSetTLVData(unsigned short Tag, unsigned char *DataIn, int DataLen);** | |
| **Function** | Set the value of the data element by specifying the tag. | |
| **Parameters** | Tag[input] | Tag of EMV standard or extended data element. |
| | DataIn[input] | The value of the data element specified by the tag. |
| | DataLen[input] | The length of the data element. |
| **Return** | EMV_OK | Succeed. |
| | EMV_DATA_ERR | Parameter error. |
| **Instruction** | The value of the standard or priority data element can be modified or set by the application. In this way, the application can satisfy the requirement in some special environment. For example, EMV library do not support blacklist check, but if the application needs to support it, there must be a function to set the value of TVR in EMV library. Normally, it’s not necessary to call this function. | |

## 2.7 EMVGetScriptResult

| | | |
|---|---|---|
| **Prototype** | **int EMVGetScriptResult(unsigned char *Result, int *RetLen);** | |
| **Function** | Gets issuer script processing result. | |
| **Parameters** | Result[output] | The issuer script processing result. |
| | RetLen[output] | The length of the issuer script processing result. |
| **Return** | EMV_OK | Succeed. |
| | EMV_NO_DATA | There is no issuer script processing result. |
| | EMV_PARAM_ERR | Parameter error. |

| | |
|---|---|
| **Instruction** | As the issuer script processing result has no relevant TLV tag, there must be a special function to get the value. |

## 2.8 EMVSetPCIModeParam

| | | |
|---|---|---|
| **Prototype** | **int EMVSetPCIModeParam(unsigned char ucPciMode, unsigned char *pucExpectPinLen, unsigned long ulTimeoutMs);** | |
| **Function** | Set whether the kernel uses PCI verify offline PIN interface or not. | |
| **Parameters** | ucPciMode[input] | 1 – Use PCI verify offline PIN interface.<br>0 – not use. |
| | pucExpectPinLen[input] | When the kernel uses PCI verify offline PIN interface, this parameter is to set the allowed length of PIN. It is a string of the enumeration of 0-12 and separated by ‘,’. For example, ‘0,4,6’ means it is allowed to input 4 or 6 digits for PIN, and to directly press Enter without input PIN. |
| | ulTimeoutMs[input] | Timeout of input PIN, unit: ms, maximum 300,000 ms. |
| **Return** | EMV_OK | Succeed. |
| | EMV_DATA_ERR | Parameter error. |
| **Instruction** | 1. The default setting is that the kernel does not use PCI verify offline PIN interface.<br>2. If setting the kernel to use PCI verify offline PIN interface, please notice the value of the 3rd parameter *pin in the call back function **cEMVGetHolderPwd**. Refer to the notes of that function for detail.<br>3. The function is Not Used for the Prolin software platform. | |

## 2.9 EMVReadVerInfo

| | | |
|---|---|---|
| **Prototype** | **int EMVReadVerInfo(char *paucVer);** | |
| **Function** | Get the version of EMV kernel and the release date. | |
| **Parameters** | paucVer[output] | Version and release date of kernel with the maximum length of 20 bytes. For example, “v26 2008.10.09”. |
| **Return** | EMV_OK | Succeed. |
| | EMV_PARAM_ERR | Parameter error. |
| **Instruction** | | |

## 2.10 EMVClearTransLog

| | | |
|---|---|---|
| **Prototype** | **int EMVClearTransLog(void);** | |
| **Function** | Clear the transaction log of kernel. | |
| **Parameters** | None | |
| **Return** | EMV_OK | Succeed. |
| **Instruction** | 1. Kernel will record the amount of the last 8 transactions after calling **EMVCoreInit**. When performing terminal risk management, kernel will add the amount of current transaction with the amount of the last transaction found in the log giving that the PAN is same and the result will be used for floor limit check. This function is provided for application to erase this log after settlement or other cases when necessary.<br>2. The function is Not Used for the Prolin software platform. | |

## 2.11 EMVAddIccTag

| | | |
|---|---|---|
| **Prototype** | **int EMVAddIccTag(ELEMENT_ATTR tEleAttr, int nAddNum);** | |
| **Function** | Add Issuer Proprietary Data Element. | |
| **Parameters** | tEleAttr[input] | List of issuer proprietary data elements. The structure adds two new variables. **Refer to appendix for structure ELEMENT _ATTR.** |
| | nAddNum[input] | Number of issuer proprietary data elements to add. |
| **Return** | EMV_OK | Succeed. |
| | EMV_PARAM_ERR | Parameter error, the number of data elements to add can't exceed 30. |
| **Instruction** | 1. Kernel only supports EMV defined standard data element. This function is used to add the list of any proprietary data elements to kernel so it will save the corresponding ones. Thus the application can access the data through **EMVGetTLVData**.<br>2. The number in the list must equal nAddNum. MaxLen and Tag must not be zero. Otherwise, parameter error will be returned.<br>3. Each time calling this function, kernel will replace the original list.<br>4. If data duplicate happens when reading ICC, the latest data will be used. | |

## 2.12 EMVSetScriptProcMethod

| | |
|---|---|
| **Prototype** | **void EMVSetScriptProcMethod (uchar ucScriptMethodIn);** |

| | | |
|---|---|---|
| **Function** | Set the special script process flag. | |
| **Parameters** | ucScriptMethodIn [input] | EMV_SCRIPT_PROC_UNIONPAY: Union pay script process method.<br>EMV_SCRIPT_PROC_NORMAL: Normal EMV script process.(Default value) |
| **Return** | NULL | |
| **Instruction** | 1. The Union Pay Network Authentication requires when there is no response with script command, the script result should be set as script not performed, while the EMVCo requires the script result should be set as script processing failed in this case.<br>2. This function is only used when there is special requirement from Union Pay Network Authentication. | |

## 2.13 EMVGetMCKParam

| | | |
|---|---|---|
| **Prototype** | **int EMVGetMCKParam(EMV_MCKPARAM *pMCKParam)** | |
| **Function** | Get the kernel MCK parameter. | |
| **Parameters** | pMCKParam[output] | Output the MCK parameter. **Refer to appendix for structure EMV_MCKPARAM.** |
| **Return** | EMV_OK | Setting successful. |
| | EMV_DATA_ERR | Parameter error. |
| **Instruction** | Before calling function **EMVSetMCKParam** to modify MCK parameter, call this function to get the MCK parameter first in order to avoid that the parameter is set to 0 by mistake. | |

## 2.14 EMVSetMCKParam

| | | |
|---|---|---|
| **Prototype** | **int EMVSetMCKParam(EMV_MCKPARAM *pMCKParam);** | |
| **Function** | Set MCK parameter. | |
| **Parameters** | pMCKParam[input] | MCK parameter.**Refer to appendix for structure EMV_MCKPARAM.** |
| **Return** | EMV_OK | Succeed |
| | EMV_DATA_ERR | Parameter is error |
| **Instruction** | Before calling this function to modify MCK parameters, please call **EMVGetMCKParam** to get the parameter. This method can avoid other parameters in this structure being changed to zero mistakely.<br>*Example:*<br>*EMV_EXTMPARAM ExTmParam;*<br>*EMV_MCKPARAM MCKParam;* | |

| | |
|---|---|
| | *memset(&ExTmParam,0,sizeof(EMV_EXTMPARAM));*<br>*memset(&MCKParam,0,sizeof(EMV_MCKPARAM));*<br>*ExTmParam.ucUseTermAIPFlg=1;*<br>*ExTmParam.aucTermAIP[0]=0x08;//Terminal is forced to perform TRM.*<br>*MCKParam.pvoid = &ExTmParam;*<br>*EMVSetMCKParam(&MCKParam);* |

## 2.15 EMVSetTmECParam

| | | |
|---|---|---|
| **Prototype** | **int EMVSetTmECParam(EMV_TMECPARAM *pParam);** | |
| **Function** | Set the terminal electronic cash related parameters. | |
| **Parameters** | pParam[input] | The electronic cash parameters which need to be set. **Refer to appendix for structure EMV_TMECPARAM.** |
| **Return** | EMV_OK | Setting successful. |
| | EMV_DATA_ERR | Parameter error. |
| **Instruction** | EMV_TMECPARAM structure specification：<br>1. Setting specification：<br>Only when the ucECTSIFlg = 1， the value of TSI and TTL can be set.<br>2. This function is used for PBOC only. | |

## 2.16 EMVSetConfigFlag

| | | |
|---|---|---|
| **Prototype** | **void EMVSetConfigFlag(int_nConfigflag);** | |
| **Function** | Set configuration. | |
| **Parameters** | nConfigflag[input] | Only bit 1(the lowest one) & bit 2 are valid at moment.<br>bit 1: 1 = support advice (default)<br>0 = not support<br>bit 2: 1= always asking user to confirm amount even no PIN input is required<br>0 = does not ask user to confirm when no PIN input is required (default)<br>bit 3: 1 = support transaction log<br>0 = not support transaction log |
| **Return** | None | |

|  |  |
|---|---|
| **Instruction** |  |

## 2.17EMVGetParamFlag

|  |  |  |
|---|---|---|
| **Prototype** | **int EMVGetParamFlag(unsigned char ucParam, int *pnFlg)** | |
| **Function** | Get the flag for signature and advice. | |
| **Parameters** | ucParam [input] | Indicate the flag to be gotten.<br>0x01 – Signature<br>0x02 – advice |
| | pnFlg[Output] | The flag got from kernel.<br>0 – NO<br>1- YES |
| **Return** | EMV_OK | Succeed. |
| | EMV_PARAM_ERR | Parameter error. |
| **Instruction** | 1. After **EMVStartTrans** and **EMVCompleteTrans** are executed, the application shall call this function to get the flag of advice, and send an advice if the flag indicates YES.<br>2. After **EMVCompleteTrans** is executed, the application shall call this function to get the flag of Signature, and require signature if the flag indicates YES. | |

# 3 Certification Authority (CA) Public Key Management

## 3.1 EMVAddCAPK

| | | |
|---|---|---|
| **Prototype** | **int EMVAddCAPK(EMV_CAPK *capk );** | |
| **Function** | Add a new CA public key. | |
| **Parameters** | capk[input] | The pointer pointing to the public key. **Refer to appendix for structure EMV_CAPK.** |
| **Return** | EMV_OK | Succeed. |
| | EMV_SUM_ERR | Checksum error. |
| | EMV_OVERFLOW | Memory overflows (EMV Kernel supports MAX_KEY_NUM CA public keys in maximum). |
| | EMV_PARAM_ERR | Parameter error. |
| **Instruction** | 1. If the public key already exists, the new key will replace the old one.<br>2. CA public key is provided by acquirer. Sometimes, the key is not assigned with the structure of EMV_CAPK. In that case, the application may need to convert the public key before it can be added to the EMV library. | |

## 3.2 EMVDelCAPK

| | |
|---|---|
| **Prototype** | **int EMVDelCAPK(unsigned char KeyID, unsigned char *RID);** |
| **Function** | Delete a CA public key. |

| | | |
|---|---|---|
| **Parameters** | KeyID[input] | The index of the key. |
| | RID[input] | Registered Application Provider Identifier. |
| **Return** | EMV_OK | Succeed. |
| | EMV_NOT_FOUND | The key does not exist. |
| | EMV_PARAM_ERR | Parameter error. |
| **Instruction** | | |

## 3.3 EMVGetCAPK

| | | |
|---|---|---|
| **Prototype** | **int EMVGetCAPK(int Index, EMV_CAPK *capk );** | |
| **Function** | Get a CA public key. | |
| **Parameters** | Index[input] | The key storage index, range(0 - MAX_KEY_NUM). |
| | capk[output] | The pointer pointing to the key. **Refer to appendix for structure EMV_CAPK.** |
| **Return** | EMV_OK | Succeed. |
| | EMV_NOT_FOUND | The key does not exist. |
| | EMV_PARAM_ERR | Parameter error. |
| **Instruction** | The application can get all the keys and then delete them. An example is like below:<br>`for(i=0;i< MAX_KEY_NUM; i ++)`<br>`{`<br>`    if(EMVGetCAPK(i, &capk )== EMV_OK)`<br>`    {`<br>`        EMVDelCAPK(capk.KeyID, capk.RID);`<br>`    }`<br>`}` | |

## 3.4 EMVCheckCAPK

| | | |
|---|---|---|
| **Prototype** | **int EMVCheckCAPK(unsigned char *KeyID, unsigned char *RID);** | |
| **Function** | Check the validity of the public keys. | |
| **Parameters** | *KeyID[output] | The index of the expired key. |
| | *RID[output] | The RID of the expired key. |
| **Return** | EMV_OK | All the keys are valid. |
| | EMV_KEY_EXP | One key is expired. |
| | EMV_PARAM_ERR | Parameter error |
| **Instruction** | Only one expired key will be returned when calling this function. When the function returns EMV_KEY_EXP, the application should handle it and then continually call this function until it returns EMV_OK. | |

# 4 Terminal Application List Management

## 4.1 EMVAddApp

| | | |
|---|---|---|
| **Prototype** | **int EMVAddApp(EMV_APPLIST *App);** | |
| **Function** | Add an EMV application to terminal application list. | |
| **Parameters** | App[input] | The pointer pointing to an application. **Refer to appendix for structure EMV_APPLIST.** |
| **Return** | EMV_OK | Succeed. |
| | EMV_OVERFLOW | Memory overflows. |
| | EMV_PARAM_ERR | Parameter error. |
| | EMV_FILE_ERR | File error |
| **Instruction** | If the application already exists, the new application will replace the old one. | |

## 4.2 EMVGetApp

| | | |
|---|---|---|
| **Prototype** | **int EMVGetApp(int Index, EMV_APPLIST *App);** | |
| **Function** | Get an application from the terminal application list.**Refer to appendix for structure EMV_APPLIST.** | |
| **Parameters** | Index[input] | The application storage index, range(0 - MAX_ APP _NUM) |
| | App[output] | The pointer pointing to an application. |
| **Return** | EMV_OK | Succeed. |
| | EMV_NOT_FOUND | The application does not exist. |
| | EMV_PARAM_ERR | Parameter error. |
| | EMV_FILE_ERR | File error |

| | |
|---|---|
| **Instruction** | The application can get all the applications from the list and then delete them. An example is like below:<br>`for(i=0;i< MAX_APP_NUM; i ++)`<br>`{`<br>`    if(EMVGetApp(i, &App)== EMV_OK)`<br>`    {`<br>`        EMVDelApp(App.AID, App.AidLen);`<br>`    }`<br>`}` |

## 4.3 EMVDelApp

| | | |
|---|---|---|
| **Prototype** | **int EMVDelApp(unsigned char *AID, int AidLen);** | |
| **Function** | Delete an application from the application list. | |
| **Parameters** | AID[input] | Application ID, compressed BCD, not more than 16 bytes. |
| | AidLen[input] | The length of AID, it must be bigger than 0. |
| **Return** | EMV_OK | Succeed. |
| | EMV_NOT_FOUND | The application does not exist. |
| | EMV_PARAM_ERR | Parameter error. |
| | EMV_FILE_ERR | File error |
| **Instruction** | | |

## 4.4 EMVGetFinalAppPara

| | | |
|---|---|---|
| **Prototype** | **int EMVGetFinalAppPara(EMV_APPLIST *ptEMV_APP);** | |
| **Function** | Get the parameter of current finally selected application. | |
| **Parameters** | ptEMV_APP[output] | The pointer pointing to the parameter of current finally selected application. **Refer to appendix for structure EMV_APPLIST.** |
| **Return** | EMV_OK | Succeed. |
| | EMV_NO_APP | No selected application. |
| | EMV_PARAM_ERR | Parameter error. |
| **Instruction** | | |

## 4.5 EMVModFinalAppPara

| | | |
|---|---|---|
| **Prototype** | **int EMVModFinalAppPara(EMV_APPLIST *ptEMV_APP);** | |
| **Function** | Modify the parameter of current finally selected application. | |
| **Parameters** | ptEMV_APP[input] | The pointer pointing to the parameter of the application. **Refer to appendix for** |

| | | |
|---|---|---|
| | | **structure EMV_APPLIST.** |
| **Return** | EMV_OK | Succeed. |
| | EMV_NO_APP | No selected application. |
| | EMV_NOT_FOUND | Specified application not found. |
| | EMV_PARAM_ERR | Parameter error. |
| **Instruction** | | |

## 4.6 EMVGetLabelList

| | | |
|---|---|---|
| **Prototype** | **int EMVGetLabelList(APPLABEL_LIST *ptAppLabel, int * pnAppNum);** | |
| **Function** | Get the label list of the application candidate list. | |
| **Parameters** | ptAppLabel[output] | The label list. **Refer to appendix for structure APPLABEL_LIST.** |
| | pnAppNum[output] | The number of application in the candidate. |
| **Return** | EMV_OK | Succeed. |
| | EMV_NO_DATA | No data available. |
| | EMV_PARAM_ERR | Parameter error. |
| **Instruction** | The terminal application can call this function in cEMVWaitAppSel to get the label list corresponding to current application candidate. | |

## 4.7 EMVDelAllApp

| | | |
|---|---|---|
| **Prototype** | **int EMVDelAllApp(void);** | |
| **Function** | Remove all application from terminal application list. | |
| **Parameters** | Null | |
| **Return** | EMV_OK | Succeed. |
| **Instruction** | | |

# 5 Terminal Revoked Issuer Public Key Certification List Management

## 5.1 EMVAddRevocList

| | | |
|---|---|---|
| Prototype | **int EMVAddRevocList(EMV_REVOCLIST *pRevocList );** | |
| Function | Add a revoked issuer public key certification to revoked certification list. | |
| Parameters | pRevocList[input] | The pointer pointing to the revoked issuer public key certification. **Refer to appendix for structure EMV_REVOCLIST.** |
| Return | EMV_OK | Succeed. |
| | EMV_DATA_ERR | Input data error (the pointer is null). |
| | EMV_OVERFLOW | Memory overflows (Kernel supports 30 revoked issuer public key certifications in maximum). |
| Instruction | 1. If the revoked issuer public key certification already exists, this function directly returns succeeding.<br>2. The structure of the revoked issuer public key certification. | |

## 5.2 EMVDelRevocList

| | | |
|---|---|---|
| Prototype | **int EMVDelRevocList(unsigned char ucIndex, unsigned char *pucRID);** | |
| Function | Delete a revoked issuer public key certification. | |
| Parameters | ucIndex[input] | The corresponding CA public key index of |

| | | |
|---|---|---|
| | | the revoked issuer public key certification. |
| | pucRID[input] | The corresponding RID of the revoked issuer public key certification. |
| **Return** | EMV_OK | Succeed. |
| | EMV_DATA_ERR | Input data error(the pointer is null) |
| | EMV_NOT_FOUND | The specified certification not found. |
| **Instruction** | | |

## 5.3 EMVDelAllRecovList

| | | |
|---|---|---|
| **Prototype** | **void EMVDelAllRecovList(void);** | |
| **Function** | Delete all revoked issuer public key certifications. | |
| **Parameters** | None | |
| **Return** | None | |
| **Instruction** | | |

# 6 Transaction Processing

EMV transaction processing functions are described in this part.

## 6.1 EMVInitTLVData

| | | |
|---|---|---|
| **Prototype** | **void EMVInitTLVData(void);** | |
| **Function** | Initialize the EMV kernel data element storage structure. | |
| **Parameters** | None | |
| **Return** | None | |
| **Instruction** | The function need to be called before the start of every transaction, for instance, before the application selection, to initialize the EMV kernel data element storage structure.<br>This function is used for contactless PBOC only. | |

## 6.2 EMVAppSelect

| | | |
|---|---|---|
| **Prototype** | **int EMVAppSelect(int Slot, unsigned long TransNo);** | |
| **Function** | EMV application selection. | |
| **Parameters** | Slot[input] | Card slot number. |
| | TransNo[input] | The sequence number of the transaction. |
| **Return** | EMV_OK | Succeed. |
| | ICC_RESET_ERR | IC card reset failed. |
| | ICC_CMD_ERR | IC card command failed. |
| | ICC_BLOCK | IC card has been blocked. |
| | EMV_NO_APP | There is no IC card application supported by |

| | | |
|---|---|---|
| | | terminal. |
| | EMV_APP_BLOCK | The EMV application has been blocked. |
| | EMV_DATA_ERR | IC card data format error. |
| | EMV_TIME_OUT | Application selection timeout. |
| | EMV_USER_CANC | Application selection is canceled by user. |
| | ICC_RSP_6985 | IC card responses with 6985 when GPO. |
| | EMV_FILE_ERR | File error |
| **Instruction** | 1. Before calling this function, the EMV IC card must be in the specified card slot which can be detected.<br>2. According to the EMV specification, if it returns EMV_NO_APP or EMV_DATA_ERR， application must prompt for 'Swipe card'.(The application can decide how to do according to the actual requirement)<br>3. EMV_USER_CANCEL and EMV_TIME_OUT are returned by callback function **cEMVWaitAppSel**. If the function **cEMVWaitAppSel** doesn't return these two values, the function **EMVAppSelect** does not return them either.(Refer to **cEMVWaitAppSel**)<br>4. ICC_RSP_6985 is returned when GPO responses 6985 and there's no application left in the application candidate list, the terminal application should then decide whether to terminate the transaction or to fallback. | |

## 6.3 EMVReadAppData

| | | |
|---|---|---|
| **Prototype** | **int EMVReadAppData(void);** | |
| **Function** | Read the selected application's data, transaction amount, etc. | |
| **Parameters** | None | |
| **Return** | EMV_OK | Succeed. |
| | ICC_CMD_ERR | IC card command failed. |
| | EMV_RSP_ERR | IC card response code error. |
| | EMV_DATA_ERR | IC card data format error. |
| | EMV_TIME_OUT | Input amount timeout. |
| | EMV_USER_CANC | Input amount is canceled by user. |
| **Instruction** | 1. EMV_USER_CANCEL and EMV_TIME_OUT are returned by callback function **cEMVInputAmount**. If the function **cEMVInputAmount** doesn't return these two values, this function does not return them either.(Refer to **cEMVInputAmount**)<br>2. If this function does not return EMV_OK, the transaction should be terminated. | |

## 6.4 EMVCardAuth

| | | |
|---|---|---|
| **Prototype** | **int EMVCardAuth(void);** | |
| **Function** | IC card data authentication. | |
| **Parameters** | None | |
| **Return** | EMV_OK | Complete. |
| | ICC_CMD_ERR | IC card command failed. |
| | EMV_RSP_ERR | IC card response code error. |
| **Instruction** | 1. This function returning EMV_OK does not stand for data authentication succeeded. The application can get the result of the authentication through function **EMVGetTLVData** and query the value of TVR. If the authentication method is CDA, this function just recovers the IC card private key. The authentication will not be done until performing Generate AC command.<br>2. If this function does not return EMV_OK, the transaction should be terminated. | |

## 6.5 EMVProcTrans

| | | |
|---|---|---|
| **Prototype** | **int EMVProcTrans(void);** | |
| **Function** | Process EMV transaction. | |
| **Parameters** | None | |
| **Return** | EMV_OK | Succeed. |
| | ICC_CMD_ERR | IC card command failed. |
| | EMV_RSP_ERR | IC card response code error. |
| | EMV_DATA_ERR | IC card data format error. |
| | EMV_NOT_ACCEPT | Transaction cannot be accepted. |
| | EMV_DENIAL | Transaction denied. |
| | EMV_TIME_OUT | Input PIN timeout. |
| | EMV_USER_CANCEL | Transaction is canceled by user. |
| | ICC_RSP_6985 | ICC response 6985 in GAC. |
| **Instruction** | 1. EMV_USER_CANCEL and EMV_TIME_OUT are returned by callback function **cEMVGetHolderPwd**. If the function **cEMVGetHolderPwd** doesn't return these two values, this function does not return them either.(Refer to **EMVAppSelect**)<br>2. Both EMV_NOT_ACCEPT and EMV_DENIAL stand for transaction failed. According to the EMV specification, EMV_NOT_ACCEPT will be returned on condition that the service does not accepted, and EMV_DENIAL will be returned on other conditions.<br>3. If this function does not return EMV_OK or ICC_RSP_6985, the transaction should be terminated.<br>4. ICC_RSP_6985 is returned when GAC responses 6985, the | |

| | |
|---|---|
| | terminal application should then decide whether to terminate the transaction or to fallback.<br>5. The function is Not Used for the Prolin software platform. |

## 6.6 EMVAppSelectForLog

| | | |
|---|---|---|
| **Prototype** | **int EMVAppSelectForLog(int Slot, unsigned char ucFlg);** | |
| **Function** | Read the card's transaction log。 | |
| **Parameters** | Slot[input] | Card slot number. |
| | ucFlg[input] | Decide whether the block application will be added to candidate list or not.<br>0-Add, 1-Not add (The default is 0) |
| **Return** | EMV_OK | Succeed. |
| | ICC_RESET_ERR | IC card reset failed. |
| | ICC_CMD_ERR | IC card command failed. |
| | ICC_BLOCK | IC card has been blocked. |
| | EMV_NO_APP | No EMV application that terminal supported. |
| | EMV_DATA_ERR | Card data format error. |
| | EMV_TIME_OUT | Application selection timeout. |
| | EMV_USER_CANCEL | Application selection is canceled by user. |
| **Instruction** | This function is similar with the function **EMVAppSelect**. The difference between them is that this function can add the block application to candidate list base on the different parameter setting when the application selection is done. | |

## 6.7 ReadLogRecord

| | | |
|---|---|---|
| **Prototype** | **int ReadLogRecord(int RecordNo);** | |
| **Function** | Read transaction log | |
| **Parameters** | RecordNo[input] | Record number. |
| **Return** | EMV_OK | Transaction processing successful. |
| | EMV_DATA_ERR | Card data error. |
| | RECORD_NOTEXIST | The specified log record doesn't exist. |
| **Instruction** | After the application finished the application selection with function | |

| | |
|---|---|
| | **EMVAppSelectForLog**, this function can be called to read the transaction log of the selected application.<br>RecordNo, Record number begins at 1, the application can read all the transaction log by following code:<br>`Reco＝1；`<br>`while(1){`<br>`    ret=ReadLogRecord(Reco);`<br>`    if(ret==RECORD_NOTEXIST){`<br>`        if(Reco==1) return NO_TRANS_LOG;`<br>`        else break;`<br>`    }`<br>`    /*Read the specific log by the function GetLogItem. */`<br>`    Reco++;`<br>`}`<br>This function can only read the transaction log to the EMV kernel buffer. Application can read the specific log by the function **GetLogItem**, such as transaction amount, transaction time and so on. |

## 6.8 GetLogItem

| | | |
|---|---|---|
| **Prototype** | **int GetLogItem(unsigned short Tag, unsigned char *TagData, int *TagLen);** | |
| **Function** | Read the data items of transaction log(PBOC compatible EMV2)。 | |
| **Parameters** | Tag[input] | The tag of data items that need to read. |
| | TagData[output] | The return value of the data items. |
| | TagLen[output] | The return length of the value of the data items. |
| **Return** | EMV_OK | Transaction processing successful. |
| | EMV_DATA_ERR | Card data error. |
| | LOGITEM_NOTEXIST | The specified log record doesn't exist. |
| **Instruction** | For each transaction log record that was read by function **ReadLogRecord**, application can read the specific log information by this function. For instance, if want to read transaction amount, you can call the function as follows:<br>**GetLogItem(0x9F02, AmtStr, AmtStrLen);**<br>/*0x9F02 is the tag of amount（Please refer to the EMV Tag list）*/ | |

## 6.9 EMVGetCardECBalance

| | | |
|---|---|---|
| **Prototype** | **int EMVGetCardECBalance(unsigned long *plBalance);** | |
| **Function** | Read the electronic cash balance from card. | |
| **Parameters** | plBalance[output] | The electronic cash balance read from card. |
| **Return** | EMV_OK | Get data succeed. |
| | EMV_DATA_ERR | Get data failed. |
| | EMV_PARAM_ERR | Parameter error. |
| **Instruction** | This function is used for PBOC only. | |

## 6.10 EMVStartTrans

| | | |
|---|---|---|
| **Prototype** | **int EMVStartTrans (ulong ulAuthAmt, ulong ulCashBackAmt, uchar *pACType);** | |
| **Function** | Performing process restrict, cardholder verification, terminal risk management & 1st GAC. | |
| **Parameters** | ulAuthAmt[input] | This authorization amount will overwrite the value input in **cEMVInputAmount**. It must be input. |
| | ulCashBackAmt[input] | This cashback amount will overwrite the value input in **cEMVInputAmount**. It must be input. |
| | pACType[output] | Output. AC type which could be: AC_TC / AC_AAC / AC_ARQC |
| **Return** | EMV_OK | Succeed. |
| | EMV_DENIAL | Transaction denied. |
| | EMV_DATA_ERR | IC card data format error. |
| | EMV_NOT_ACCEPT | Transaction is not accepted. |
| | ICC_CMD_ERR | IC card command failed. |
| | ICC_RSP_6985 | ICC response 6985 in 1st GAC. |
| | EMV_RSP_ERR | IC card response code error. |
| | EMV_PARAM_ERR | Parameter error. |
| **Instruction** | 1. If the return code is EMV_DENIAL or EMV_NOT_ACCEPT, the application should get the CID (9F27) from kernel, and check if the advice message is needed.<br>2. If the return code is EMV_OK, pACType shall be checked. If pACType is AC_TC, the transaction shall be approved. If pACType is AC_ARQC, the transaction shall go online and then **EMVCompleteTrans** must to be called. | |

## 6.11 EMVCompleteTrans

| | | |
|---|---|---|
| **Prototype** | **int EMVCompleteTrans(int nCommuStatus, unsigned char *paucScript, int *pnScriptLen, uchar *pACType);** | |
| **Function** | Performing online data processing (issuer authentication, script processing, etc) & 2nd GAC. | |
| **Parameters** | nCommuStatus[input] | Input. Online status which could be:<br>ONLINE_APPROVE - Online approved or online referral approved.<br>ONLINE_FAILED - Online failed.<br>ONLINE_DENIAL – Online rejected or online referral rejected.<br>If the online status is out of these ranges, the application should reject the transaction. |
| | *paucScript[input] | Input. Issuer script data in TLV format. |
| | *pnScriptLen[input] | Input. The length of issuer script data. |
| | pACType[output] | Output: AC type which could be:<br>AC_TC Transaction approved<br>AC_AAC Terminal requests TC, but rejected by ICC.<br>AC_AAC_HOST Transaction rejected by issuer. |
| **Return** | EMV_OK | Succeed. |
| | EMV_DENIAL | Transaction denied. |
| | EMV_DATA_ERR | IC card data format error. |
| | EMV_NOT_ACCEPT | Transaction is not accepted. |
| | ICC_CMD_ERR | IC card command failed. |
| | ICC_RSP_6985 | ICC response 6985 in 2nd GAC. |
| | EMV_RSP_ERR | IC card response code error. |
| | EMV_PARAM_ERR | Parameter error. |
| **Instruction** | 1. The following data element, if exists, must be sent to kernel through EMVSetTLVData before calling this function:<br>ARC – 8A<br>AC – 89<br>IAD – 91<br>2. If the return code is EMV_DENIAL or EMV_NOT_ACCEPT, Application should get the CID (9F27) from kernel, and check if the advice message is needed.<br>3. If the return code is EMV_DENIAL, and the transaction is online data captured, and the online result is ONLINE_APPROVE, then application shall send a reversal message to host.<br>4. If the return code is not EMV_OK, **EMVGetScriptResult** must be called, and if **EMVGetScriptResult** returns EMV_OK, Application should send a advice message if advice is supported. | |

## 6.12 EMVSwitchClss

| | | |
|---|---|---|
| **Prototype** | **int EMVSwitchClss(Clss_TransParam *ptTransParam,uchar *pucSelData, int nSelLen, uchar *pucGPOData, int nGPOLen);** | |
| **Function** | Set response data of final selection and GPO command into EMV kernel. | |
| **Parameters** | ptTransParam[input] | Transaction related parameters. Please refer to the contactless common data instruction documentation for the parameter structure.<br>**Refer to appendix for structure Clss_TransParam.** |
| | pucSelData[input] | Response data of final selection command.,which need to be got by **Clss_GetFinalSelectData_Entry** in Entry library. |
| | nSelLen[input] | The data length return by final selection command, which need to be got by **Clss_GetFinalSelectData_Entry** in Entry library. |
| | pucGPOData[input] | The data of GPO command and response, which need to be got by **Clss_GetGPOData_Pboc** in qPBOC library. |
| | nGPOLen[input] | The data length of GPO command and response, which need to be got by **Clss_GetGPOData_Pboc** in qPBOC library. |
| **Return** | EMV_OK | Succeed. |
| | EMV_PARAM_ERR | Parameter error. |
| | EMV_NO_APP | There is no application at candidate list. |
| | EMV_DATA_ERR | Data error. |
| **Instruction** | This function is only used for contactless PBOC application. | |

## 6.13 EMVSetAmount

| | | |
|---|---|---|
| **Prototype** | **int EMVSetAmount(unsigned char * szAuthAmount, unsigned char * szCashBackAmount);** | |
| **Function** | Set transaction amount and cashback amount bigger than 0xffffffff,. | |
| **Parameters** | szAuthAmount[input] | Authorised Amount (Tag 9F02, n12) |
| | szCashBackAmount[input] | Other Amount(Tag 9F03, n12)<br>If Other Amount does not exist, the value should be NULL. |
| **Return** | EMV_OK | Succeed. |
| | EMV_DATA_ERR | Data error. |
| **Instruction** | If the amount exceeds 0xffffffff, please use this function to set the amount to kernel.<br>Please call this function before **EmvProcTrans** or **EmvStartTrans**.<br>If use this function to set amount, please make **cEMVInputAmount** return two zero amount.<br>For example:<br>`cEMVInputAmount（unsigned long *AuthAmt, unsigned long *CashBackAmt）`<br>`{`<br>`    // if transaction type = CashBack`<br>`    *AuthAmt = 0;`<br>`    *CashBackAmt = 0;`<br>`    return 0;`<br>`}`<br>Note: If the amount is smaller than 0xffffffff, this amount will be overwrited by the amount input in **EMVStartTrans**. | |

## 6.14 EMVReadSingleLoadLog

| | | |
|---|---|---|
| **Prototype** | **int EMVReadSingleLoadLog (int nRecordNoIn);** | |
| **Function** | Read single loading log (PBOC3.0). | |
| **Parameters** | nRecordNoIn[input] | Record number. |
| **Return** | EMV_OK | Transaction processing successful. |
| | EMV_DATA_ERR | Card data error. |

| | | |
|---|---|---|
| | RECORD_NOTEXIST | The specified log record doesn't exist. |
| **Instruction** | This function can be called to read the loading log of the selected application. And it should be called after the function **EMVAppSelectForLog**.<br>RecordNo, Record number begins at 1, the application can read all the loading logs by following code:<br>`Reco＝1；`<br>`while(1){`<br>`ret=EMVReadSingleLoadLog (Reco);`<br>`if(ret==RECORD_NOTEXIST){`<br>`if(Reco==1) return NO_TRANS_LOG;`<br>`else break;`<br>`}`<br>`/*Read the specific log by the function EMVGetSingleLoadLogItem. */`<br>`Reco++;`<br>`}`<br>This function only save the loading log to the EMV kernel buffer. Application could read the specific log item by the function **EMVGetSingleLoadLogItem**, such as transaction amount, transaction time and so on. | |

## 6.15 EMVGetSingleLoadLogItem

| | | |
|---|---|---|
| **Prototype** | **int EMVGetSingleLoadLogItem(unsigned short usTagIn, unsigned char *paucDataOut, int *pnLenOut);** | |
| **Function** | Read the data item of loading log (PBOC3.0). | |
| **Parameters** | usTagIn[input] | The tag of data item to be read. |
| | paucDataOut[output] | Pointer to the value of the data item. |
| | pnLenOut[output] | Pointer to the length of data item. |
| **Return** | EMV_OK | Transaction processing successful. |
| | EMV_DATA_ERR | Card data error. |
| | LOGITEM_NOTEXIST | The specified log record doesn't exist. |
| | NO_TRANS_LOG | The entry of loading log doesn't exist. |
| **Instruction** | For each loading log record that was read by function **EMVReadSingleLoadLog**, application can read the specific log information by this function. For example, if want to read save transaction amount, you can call the function as follows:<br>**EMVGetSingleLoadLogItem(0x9F02, AmtStr, AmtStrLen);** | |

| | |
|---|---|
| | /*0x9F02 is the tag of amount（Please refer to the EMV Tag list）*/ |

## 6.16 EMVReadAllLoadLogs

| | | |
|---|---|---|
| **Prototype** | **int EMVReadAllLoadLogs(unsigned char *paucLogDataOut, int *pnLenOut);** | |
| **Function** | Read all loading logs (PBOC3.0). | |
| **Parameters** | paucLogDataOut[output] | Pointer to the loading log data. |
| | pnLenOut[output] | Pointer to the length of the loading log data. |
| **Return** | EMV_OK | Transaction processing successful. |
| | EMV_DATA_ERR | Card data error. |
| | RECORD_NOTEXIST | The specified log record doesn't exist. |
| **Instruction** | This function can read out all the loading transaction logs in one time, and saves them into the EMV kernel buffer. Please refer to table 12 and 13 in PBOC 3.0 book 13 for the format of the loading log. | |

## 6.17 EMVGetLogData

| | | |
|---|---|---|
| **Prototype** | **int EMVGetLogData (unsigned char *paucLogDataOut, int *pnLenOut);** | |
| **Function** | Get the log data which is read from card. | |
| **Parameters** | paucLogDataOut[output] | Pointer to the log data. |
| | pnLenOut[output] | Pointer to the length of the log data. |
| **Return** | EMV_OK | Transaction processing successful. |
| | EMV_PARAM_ERR | The parameter is error. |
| | NO_TRANS_LOG | There is no log data. |
| **Instruction** | 1. If the application read loading log before calling this function, the log data which is got by this function is loading log data. Please refer to PBOC 3.0 Book 13 for the format of the loading log data.<br>2. If the application read transaction log before calling this function, the log data which is got by this function is transaction log data. Please refer to PBOC 3.0 Book 13 for the format of the transaction log data. | |

# 7 Callback Functions

The interface of the callback functions which kernel needs to call are described in this part. The terminal application needs to implement these functions.

## 7.1 cEMVWaitAppSel

| | | |
|---|---|---|
| **Prototype** | **int cEMVWaitAppSel(int TryCnt, EMV_APPLIST List[], int AppNum);** | |
| **Function** | Wait for user to select an application from the application candidate list. If there is only one application in the application list and it doesn't require cardholder confirmation, this function will not be called. | |
| **Parameters** | TryCnt[input] | TryCnt=0 means it is called for the first time, otherwise, it has been called more than one time. (According to the EMV specification, if this function has been called more than one time, terminal should prompt for 'APP NOT ACCEPTS, TRY AGAIN' or some other word like that.). |
| | List[][input] | Application candidate list. **Refer to appendix for the structure of EMV_APPLIST.** |
| | AppNum[input] | The number of the application in the list. |
| **Return** | >=0 | The sequence number selected by the user (For example: 0 stands for List[0] was selected). |
| | EMV_USER_CANCEL | Application selection is canceled by user. |

| | | |
|---|---|---|
| | EMV_TIME_OUT | Application selection timeout. |
| **Instruction** | The application is listed in the sequence according to the EMV specification. As the List is also used for output parameter, so if there is any modification to the list, directly operate to the pointer.<br>Refer to **EMVAppSelect** for the purpose of EMV_USER_CANCEL and EMV_TIME_OUT. | |

## 7.2 cEMVInputAmount

| | | |
|---|---|---|
| **Prototype** | **int cEMVInputAmount(unsigned long *AuthAmt, unsigned long *CashBackAmt);** | |
| **Function** | Input transaction amount. | |
| **Parameters** | AuthAmt[output] | Transaction amount. |
| | CashBackAmt[output] | Cashback amount. If CashBackAmt is NULL, there is no need to input the amount. If CashBackAmt is not NULL, input the amount or set it as 0. |
| **Return** | EMV_OK | Input succeeds. |
| | EMV_USER_CANCEL | Amount input is canceled by user. |
| | EMV_TIME_OUT | Amount input timeout. |
| **Instruction** | Refer to **EMVAppSelect** for the purpose of EMV_USER_CANCEL and EMV_TIME_OUT. | |

## 7.3 cEMVGetHolderPwd

| | | |
|---|---|---|
| **Prototype** | **int cEMVGetHolderPwd(int TryFlag, int RemainCnt, char *pin);** | |
| **Function** | Wait for cardholder to input PIN. | |
| **Parameters** | TryFlag[input] | 0: It's the first time calling this function to get the cardholder's PIN in this transaction.<br>1: It's not the first time calling this function to get the cardholder's PIN in this transaction. (It appears only when verifying the offline PIN and failing)<br>2: PIN is not required, calling this function to allow the cardholder to confirm the amount only. RemainCnt is set as 0 and pin is set as NULL. |

| | | |
|---|---|---|
| | RemainCnt[input] | 1. The chance remained to verify the PIN. If RemainCnt equals 1, it means only one chance remained to verify the PIN, and if the following PIN verification is still failed, the PIN will be blocked.<br>2. If the parameter pin is NULL, it means to input online PIN. In this case, TryFlag and RemainCnt don’t make sense; the PIN will be encrypted and stored by the terminal application and send to host when online, and then verified by the host. |
| | Pin[input/output] | 1. If PCI verify offline PIN interface is not used, this parameter is output. The application should return plaintext PIN in it with string ended with ’\x00’.<br>2. If PCI verify offline PIN interface is used, this parameter is input. The application doesn’t need to return anything in it. The possible value of pin[0] in this condition could be:<br>● EMV_PED_WAIT: PIN input interval not enough.<br>● EMV_PED_TIMEOUT: PIN input timeout.<br>● EMV_PED_FAIL: PED locked or other failure. |
| **Return** | EMV_OK | Succeed. |
| | EMV_NO_PINPAD | There is no pinpad or pinpad does not work. |
| | EMV_NO_PASSWORD | No PIN or cardholder doesn’t want to input PIN. |
| | EMV_USER_CANCEL | PIN input is canceled by user. |
| | EMV_TIME_OUT | PIN input timeout. |
| **Instruction** | 1. The application can flexibly arrange PIN input interface which fulfill user’s requirement. The mode of PIN input can be freely controlled by the application, at the same time it must be compatible with the EMV security specification.<br>2. Refer to **EMVProcTrans** for the purpose of EMV_USER_CANCEL and EMV_TIME_OUT.<br>3. From version V25_T4, if this function is used for PCI compliant products such as S80/S90/SP30, interface **EMVSetPCIModeParam** should be used to set the kernel using PCI verify offline PIN interface or not.<br>4. In monitor platform, if setting kernel to use PCI verifies offline PIN interface. In prolin platform, the operation of offline PIN is completed by callback function **cEMVPedVerifyCipherPin**. This function does not need to implement the operation of PIN | |

| | |
|---|---|
| | input, but still need to display the prompt information to cardholder such as transaction amount and "Please input PIN".<br>5. If setting the kernel to use PCI verify offline PIN interface, when input PIN fail, the 1st byte of the 3rd parameter *pin is used to carry the failure information. The possible values of pin[0] include EMV_PED_WAIT, EMV_PED_TIMEOUT and EMV_PED_FAIL. |

## 7.4 cEMVReferProc

| | | |
|---|---|---|
| **Prototype** | **int cEMVReferProc(void);** | |
| **Function** | Process referral activated by the issuer. | |
| **Parameters** | None | |
| **Return** | REFER_APPROVE | Referral approved. |
| | REFER_DENIAL | Referral denied. |
| **Instruction** | 1. For the referral activated by the issuer, the function only returns REFER_DENIAL or REFER_APPROVE.<br>2. If the acquirer doesn't support referral processing, the application can directly return REFER_DENIAL according to the acquirer's requirement.<br>3. The function is Not Used for the Prolin software platform. | |

## 7.5 cEMVOnlineProc

| | | |
|---|---|---|
| **Prototype** | **int cEMVOnlineProc(unsigned char *RspCode, unsigned char *AuthCode, int *AuthCodeLen, unsigned char *IAuthData, int *IAuthDataLen, unsigned char *script, int *ScriptLen);** | |
| **Function** | Online transaction. | |
| **Parameters** | RspCode[output] | Authorization response code, 2 bytes. Set RspCode[0] as 0 in case of online failed. |
| | AuthCode[output] | Authorization code，6 bytes. |
| | AuthCodeLen[output] | The length of the AuthCode. Set it as 0 when there is no AuthCode. |
| | IAuthData[output] | Issuer authentication data returned from host. |
| | IAuthDataLen[output] | The length of the IAuthData. Set it as 0 when there is no IAuthData. |
| | Script[output] | Issuer script. If the scripts are not sent in one 8583 field, then put all the scripts together and return by this parameter. |
| | ScriptLen[output] | The length of the script. Set it as 0 when there is no script. |
| **Return** | ONLINE_APPROVE | Online transaction approved (host approve the transaction). |

| | | |
|---|---|---|
| | ONLINE_DENIAL | Online transaction denied (host denied the transaction). |
| | ONLINE_REFER | Online transaction referral (Issuer referral). |
| | ONLINE_FAILED | Online transaction failed. |
| | ONLINE_ABORT | Online transaction aborted (PBOC requirement). |
| **Instruction** | 1. This function processes as below:<br>Firstly, it gets the IC card transaction data required by online processing and assembles data package;<br>Secondly, it connects the host to perform data communication.<br>At last, it returns the data received from host to the EMV library through the parameter.<br>If online failed, for example, connection failed or receiving data failed, it must return ONLINE_FAILED.<br>The application need to handle automatic reversal by itself.<br>2. The function is Not Used for the Prolin software platform. | |

## 7.6 cEMVAdviceProc

| | | |
|---|---|---|
| **Prototype** | **void cEMVAdviceProc(void);** | |
| **Function** | Online or offline advice processing. | |
| **Parameters** | None | |
| **Return** | None | |
| **Instruction** | 1. Whether supporting advice processing or not is determined by the acquirer (or issuer). Directly return if it does not support.<br>2. Whether to send the data package to the host immediately or store it for later sending online is determined by the acquirer (or issuer).<br>3. The application can get the data in the advice data package which may be useful through the function **EMVGetTLVData**. | |

## 7.7 cEMVVerifyPINOK

| | | |
|---|---|---|
| **Prototype** | **void cEMVVerifyPINOK(void);** | |
| **Function** | Prompt for "PIN OK". | |
| **Parameters** | None | |
| **Return** | None | |
| **Instruction** | 1. When EMV library verify PIN failed, it will call the function **cEMVGetHolderPwd** to prompt for error massage.<br>2. When EMV library verifies PIN and passes, it will call this function. The terminal application may prompt for 'PIN OK' in this function. If this is not necessary, the function can directly return. | |

## 7.8 cEMVUnknowTLVData

| | | |
|---|---|---|
| **Prototype** | **int cEMVUnknowTLVData(unsigned short Tag, unsigned char *dat, int len);** | |
| **Function** | Get the data of the unknown tag. | |
| **Parameters** | Tag[input] | Tag. It may be not defined by EMV or defined by EMV but can't be found in the IC card. |
| | Dat[output] | The value of the tag, filled by the application. |
| | Len[output] | The length of the tag according to the DOL requirement. |
| **Return** | -1 | The application offers nothing for the tag. |
| | 0 | The application offers the value of the tag. |
| **Instruction** | 1.For the Prolin software platform, the kernel will obtain the following Tags' value by this function:<br>Unpredictable Number (9F37)<br>Transaction Date (9A).<br>Transaction Time (9F21)<br>(IFD) Serial Number (9F1E)<br>Last transaction amount of current PAN & PANSN (FF01)<br>2. If EMV library doesn't recognize a tag when processing DOL, it will call this callback function and require application to return the value of the tag. If the application cannot provide, it can directly return -1.This function is also provided to satisfy the special application which is out of scope of EMV requirement. It returns -1 normally. | |

## 7.9 cCertVerify

| | | |
|---|---|---|
| **Prototype** | **int cCertVerify(void);** | |
| **Function** | Cardholder credential verify(PBOC) | |
| **Parameters** | None | |
| **Return** | -1 | Verify failed. |

| | | |
|---|---|---|
| | 0 | Verify succeeded. |
| **Instruction** | PBOC adds the CVM, which is for cardholder credential verify. During the transaction, when the EMV kernel finds that the current used PBOC card need to do the cardholder verify, it will call this function. The application should call the function **EMVGetTLVData** to read the credential number and credential type, and provide the relevant information to the operator.<br>This function is only used for PBOC. | |

## 7.10 cEmvSetParam

| | | |
|---|---|---|
| **Prototype** | **int cEmvSetParam(void);** | |
| **Function** | Set parameter after application selection. | |
| **Parameters** | None | |
| **Return** | EMV_OK | Succeed. |
| | Others | Abort/Terminate current transaction. |
| **Instruction** | 1. This function is used to set some AID specific parameter after performing application selection and before GPO. Application can call **EMVSetTLVData** in this function to set these parameters.<br>2. When the return value of this function is not EMV_OK, kernel will abort the current transaction.<br>3. If the terminal support SM, application shall set DF69 with value 1, length 1 in this function. | |

## 7.11 cEMVPiccIsoCommand

| | | |
|---|---|---|
| **Prototype** | **unsigned char cEMVPiccIsoCommand (unsigned char cid,APDU_SEND *ApduSend,APDU_RESP *ApduRecv);** | |
| **Function** | Data exchange between the contactless reader and the contactless card. | |
| **Parameters** | cid[input] | Specify the logical channel of card. The logical channel is output by the CID parameter of PiccDetect(). The channel number range from 0 to 14, and the current values are 0. |
| | ApduSend [input] | The command data structure send to PICC.(**Refer to appendix for the data structure of APDU_SEND** ) |

| | | |
|---|---|---|
| | ApduRecv [output] | The data structure received by PICC.( **Refer to appendix for the data structure of APDU_RESP**) |
| **Return** | Please refer to Products Application Developer Guide of PAX for the PICC operation. The kernel only cares about the PICC operation succeed or not. If succeed, return 0x00, others return 0x01. | |
| **Instruction** | 1. This function is only used for contactless PBOC. If the kernel is only used for the contact application, you can set this callback function to an empty function.<br>2. Parameters cid and pucSelData are controlled by application. The kernel only uses the output parameter and return value. | |

## 7.12 cEMVIccIsoCommand

| | | |
|---|---|---|
| **Prototype** | **uchar cEMVIccIsoCommand(uchar slot,<br>APDU_SEND *ApduSend,<br>APDU_RESP *ApduRecv);** | |
| **Function** | IC card operating function. | |
| **Parameters** | Slot [Input] | Specify the logical channel of card. The logical channel is output by **IccDetect** and its current values are 0. |
| | ApduSend [Input] | The command data structure send to ICC.(**Refer to appendix for the data structure of APDU_SEND** ) |
| | ApduRecv [Output] | The data structure received by ICC.( **Refer to appendix for the data structure of APDU_RESP**) |
| **Return** | Please refer to products Application Developer Guide of PAX for the ICC operation. The kernel only cares about the ICC operation succeed or not. If succeed, return 0x00, others return 0x01. | |
| **Instruction** | 1. Interactive with IC card to get the relevant information of command.<br>2. The function is only used for the Prolin software platform.<br>3. Parameters slot and ApduSend are controlled by application. The kernel only uses the output parameter and return value. | |

## 7.13 cEMVPedVerifyPlainPin

| | | |
|---|---|---|
| **Prototype** | **int cEMVPedVerifyPlainPin (uchar IccSlot,<br>uchar *ExpPinLenIn,<br>uchar *IccRespOut,<br>uchar Mode,<br>ulong TimeoutMs);** | |
| **Function** | Get offline plaintext PIN and verify offline plaintext PIN. | |
| **Parameters** | IccSlot [Input] | IC card slot. |
| | ExpPinLenIn [Input] | The input string which has the valid length. |
| | IccRespOut [Output] | The status code of IC card response. |
| | Mode [Input] | IC card command mode. |
| | TimeoutMs [Input] | The timeout for enter PIN. |
| **Return** | Application need to transform the return value to the following six return values, which can refer to products Application Developer Guide of PAX:<br>PED_RET_ERR_NO_PIN_INPUT<br>PED_RET_ERR_INPUT_CANCEL<br>PED_RET_ERR_ICC_NO_INIT<br>PED_RET_ERR_NO_ICC<br>PED_RET_ERR_WAIT_INTERVAL<br>PED_RET_OK<br>Other return values can return without any transform. | |
| **Instruction** | 1. The function is only used for the Prolin software platform.<br>2. Parameters ExpPinLenIn and TimeoutMs are controlled by the application, other parameters are input by EMV kernel.<br>3. If there is an external PINPAD, the application should construct the PIN block and do PIN verify in this callback function. Please refer to section 6.5.12 of EMV book 3 for PIN block construction and verification. | |

## 7.14 cEMVPedVerifyCipherPin

| | | |
|---|---|---|
| **Prototype** | **int cEMVPedVerifyCipherPin (uchar IccSlot,**<br>**uchar *ExpPinLenIn,**<br>**RSA_PINKEY *RsaPinKeyIn,**<br>**uchar *IccRespOut,**<br>**uchar Mode,**<br>**ulong TimeoutMs);** | |
| **Function** | Get offline enciphered PIN and verify offline enciphered PIN. | |
| **Parameters** | IccSlot [Input] | IC card slot. |
| | ExpPinLenIn [Input] | The input string which has the valid length. |
| | RsaPinKeyIn [Input] | The data structure required by encryption.(**Refer to the appendix for the data structure RSA_PINKEY**) |
| | Mode [Input] | IC card command mode. |
| | IccRespOut [Output] | The status code of IC card response. |
| | TimeoutMs [Input] | The timeout for enter PIN. |
| **Return** | Application need to transform the return value to following six return values, which can refer to products Application Developer Guide of PAX:<br>PED_RET_ERR_NO_PIN_INPUT<br>PED_RET_ERR_INPUT_CANCEL<br>PED_RET_ERR_ICC_NO_INIT<br>PED_RET_ERR_NO_ICC<br>PED_RET_ERR_WAIT_INTERVAL<br>PED_RET_OK<br>Other return values can return without any transform. | |
| **Instruction** | 1. The function is only used for the Prolin software platform.<br>2. Parameters ExpPinLenIn and TimeoutMs are controlled by the application, other parameters are input by EMV kernel.<br>3. If there is an external PINPAD, the application should construct the PIN block, PIN encrypts and verification in this callback functions.<br>Please refer to section 6.5.12 of EMV book 3 for PIN block construction and verification. | |

| | |
|---|---|
| | Please refer to section 7.2 of EMV book 2 for PIN encrypt. |

## 7.15 cEMVSM2Verify

| | | |
|---|---|---|
| **Prototype** | **unsigned char cEMVSM2Verify(unsigned char *paucPubkeyIn,unsigned char *paucMsgIn,int nMsglenIn, unsigned char *paucSignIn, int nSignlenIn);** | |
| **Function** | SM2 verification. | |
| **Parameters** | paucPubkeyIn [input] | Pointer to the public key. |
| | paucMsgIn [input] | Pointer to the message buffer. |
| | nMsglenIn [input] | Length of the message. |
| | paucSignIn [input] | Pointer to the signature of the message. |
| | nSignlenIn [input] | Length of the signature of the message. |
| **Return** | EMV_OK | Success |
| | Other | Fail |
| **Instruction** | Example:<br> #define ENTLA "\x00\x80"<br>#define IDA "\x31\x32\x33\x34\x35\x36\x37\x38\x31\x32\x33\x34\x35\x36\x37\x38"<br><br>unsigned char cEMVSM2Verify(unsigned char *paucPubkeyIn,unsigned char *paucMsgIn,int nMsglenIn, unsigned char *paucSignIn, int nSignlenIn)<br>{<br>Gen_Za(ENTLA, 2, IDA, 16, aucPubKey);<br>return SM2_Verify(aucPubKey, aucInputData, nInputLen, aucSigData, nSigLen);<br>} | |

## 7.16 cEMVSM3

| | | |
|---|---|---|
| **Prototype** | **unsigned char cEMVSM3(unsigned char * paucMsgIn, int nMsglenIn,unsigned char * paucResultOut);** | |
| **Function** | SM3 hash calculation. | |
| **Parameters** | paucMsgIn [Input] | Pointer to message buffer. |

| | | |
|---|---|---|
| | paucMsgIn [Input] | Pointer to length of message. |
| | paucResultOut [Output] | Pointer to output hash result buffer. |
| **Return** | EMV_OK | Success |
| **Instruction** | Example:<br>unsigned char cEMVSM3(unsigned char * paucMsgIn, int nMsglenIn,unsigned char * paucResultOut)<br>{<br>    sm3(input,ilen,output);<br>    return EMV_OK;<br>} | |

# 8 Debug Interface

## 8.1 EMVSetDebug

| | | |
|---|---|---|
| **Prototype** | **void EMVSetDebug(int EnableFlag);** | |
| **Function** | Enable/Disable debug state. | |
| **Parameters** | EnableFlag[input] | 1 — Enable debug state.<br>0 — Disable debug state. |
| **Return** | None | |
| **Instruction** | 1. After enabling debug state, kernel will send all the commands and data which it sent to and received from the IC card through COM1 with the baud rate of 115200, 8, n, 1. Then developer can receive them and analyze by hyper terminal or other tools.<br>The default debug state of kernel is 0 (Disabled).<br>2. The function is Not Used for the Prolin software platform. | |

## 8.2 EMVGetICCStatus

| | | |
|---|---|---|
| **Prototype** | **void EMVGetICCStatus(unsigned char *SWA, unsigned char *SWB);** | |
| **Function** | Read the response status word of the last ICC command of a failed transaction. | |
| **Parameters** | SWA[output] | SWA. |
| | SWB[output] | SWB. |
| **Return** | None | |
| **Instruction** | The function is Not Used for the Prolin software platform. | |

## 8.3 EMVGetVerifyICCStatus

| | | |
|---|---|---|
| **Prototype** | **int EMVGetVerifyICCStatus(unsigned char *pucSWA, unsigned char *pucSWB);** | |
| **Function** | Read the response status word of the PIN Verify command. | |
| **Parameters** | pucSWA[output] | Pointer to SWA. |
| | pucSWB[output] | Pointer to SWB. |
| **Return** | EMV_OK | Successful. |
| | EMV_NO_DATA | There is no PIN verify command in this transaction. |
| | EMV_PARAM_ERR | The parameter is error. |
| **Instruction** | None | |

## 8.4 EMVGetDebugInfo

| | | |
|---|---|---|
| **Prototype** | **int EMVGetDebugInfo(int nExpAssistInfoLen, uchar *paucAssistInfo, int *pnErrorCode)** | |
| **Function** | Get the debug information. | |
| **Parameters** | **nExpAssistInfoLen [input]** | RFU |
| | **paucAssistInfo[output]** | RFU |
| | **pnErrorCode[output]** | The error code. |
| **Return** | EMV_OK | Successful. |
| | EMV_PARAM_ERR | The parameters are error. |
| **Instruction** | 1. This function is only used to get the debug information when a transaction is failed.<br>2. So far, it only can be used to get the error code of offline data authorization. | |

| **Value** | **Retrieval of Issuer Public Key Error** | **Suggestion** |
|---|---|---|
| **1** | Authority Public Key Index is not present. | Check the card data |
| **2** | Issuer Public Key Certificate is not present. | Check the card data |
| **3** | Issuer Public key Exponent is not present. | Check the card data |
| **4** | The CAPK indicated by Authority Public Key Index is not present in kernel. | Check if the CAPK indicated by Authority Public Key Index has been added in kernel by |

| | | |
|---|---|---|
| | | **EMVAddCAPK**. |
| **5** | Issuer Public Key Certificate has a length different from the length of the Certification Authority Public Key Modulus | Check CAPK data and the card data |
| **6** | Recovered Data Header is not equal to '6A', | Check the card data |
| **7** | Certificate Format is not equal to '02' | Check the card data |
| **8** | Recovered Data Trailer is not equal to 'BC' | Check the card data |
| **9** | The Length of Issuer Public Key recovered is not equal to actual length | Check the card data |
| **10** | The hash result calculated is not equal to the hash result recovered. | Check CAPK data and the card data |
| **11** | Issuer Identifier does not match the leftmost 3-8 PAN digits | Check the card data |
| **12** | The issuer public key certificate has expired | Check the card data |
| **13** | Hash Algorithm Indicator or Issuer Public Key Algorithm Indicator is not recognized | Check the card data |
| **14** | The CAPK is in the revocation list | Check the revocation list |
| | **Retrieval of ICC Public Key Error** | |
| **21** | ICC Public Key Certificate is not present. | Check the card data |
| **22** | ICC Public Key Exponent is not present. | Check the card data |
| **23** | ICC Public Key Certificate has a length different from the length of the Issuer Public Key Modulus obtained in the previous section. | Check the card data |
| **24** | Recovered Data Header is not equal to '6A', | Check the card data |
| **25** | Certificate Format　is not equal to '04' | Check the card data |
| **26** | Recovered Data Trailer is not equal to 'BC' | Check the card data |
| **27** | The Length of ICC Public Key recovered is not equal to actual length | Check the card data |

| | | |
|---|---|---|
| **28** | Static Data Authentication Tag List is present and contains tags other than '82' | Check the card data |
| **29** | The hash result calculated is not equal to the hash result recovered. | Check the card data |
| **30** | Recovered PAN is not the same as the Application PAN read from the ICC. | Check the card data |
| **31** | The ICC public key certificate has expired | Check the card data |
| **32** | ICC Public Key Algorithm Indicator is not recognized | Check the card data |
| | **SDA Verification Error** | |
| **41** | Signed Static Application Data is not present. | Check the card data |
| **42** | Signed Static Application Data has a length different from the length of the Issuer Public Key Modulus | Check the card data |
| **43** | Recovered Data Header is not equal to '6A'. | Check the card data |
| **44** | Signed Data Format is not equal to '03' | Check the card data |
| **45** | Recovered Data Trailer is not equal to 'BC' | Check the card data |
| **46** | The Static Data Authentication Tag List is present and contains tags other than '82'. | Check the card data |
| **47** | The hash result calculated is not equal to the hash result recovered. EMV Book 2, 5.4 steps 7. | Check the card data |
| | **CDA Verification Error** | |
| **61** | Signed Dynamic Application Data is not present. | Check the card data |
| **62** | Signed Dynamic Application Data has a length different from the length of the ICC Public Key Modulus | Check the card data |
| **63** | Recovered Data Header is not equal to '6A'. | Check the card data |

| | | |
|---|---|---|
| **64** | Signed Data Format is not equal to '05' | Check the card data |
| **65** | Recovered Data Trailer is not equal to 'BC' | Check the card data |
| **66** | Cryptogram Information Data retrieved from the ICC Dynamic Data is not the same as the Cryptogram Information Data obtained from the response to the GENERATE AC command. | Check the card data |
| **67** | The hash result calculated is not equal to the hash result recovered. EMV Book 2, 6.6.2 step 9. | Check the card data |
| **68** | The hash result calculated is not equal to the hash result recovered. EMV Book 2, 6.6.2 step 12. | Check the card data |
| | **DDA Verification Error** | |
| **81** | The DDOL in the ICC does not include the Unpredictable Number. | Check the card data |
| **82** | The ICC does not contain a DDOL and the terminal does not contain a default DDOL. | The application shall set a default DDOL into kernel. |
| **83** | The ICC does not contain a DDOL and the default DDOL in the terminal does not include the Unpredictable Number. | Check the default DDOL of terminal. |
| **84** | DDOL related data filling error. | Check the data required in DDOL |
| **85** | INTERNAL AUTHENTICATE command send or receive error | Check the return code of callback function to send the command. |
| **86** | The TLV format of response data of INTERNAL AUTHENTICATE is error. | Check the response data of the command. |
| **87** | The response data of INTERNAL AUTHENTICATE is not ‘80’ or ‘77’ | Check the card data |
| **88** | There is no data in template ‘80’. | Check the card data |
| **89** | The length of ‘80’ is error. | Check the card data |

| | | |
|---|---|---|
| **90** | There is no Signed Dynamic Application Data in the response data of INTERNAL AUTHENTICATE | Check the card data |
| **91** | The Signed Dynamic Application Data has a length different from the length of the ICC Public Key Modulus | Check the card data |
| **92** | Recovered Data Header is not equal to '6A'. | Check the card data |
| **93** | Signed Data Format is not equal to '05' | Check the card data |
| **94** | Recovered Data Trailer is not equal to 'BC' | Check the card data |
| **95** | The hash result calculated is not equal to the hash result recovered. EMV Book 2, 6.5.2 step 7. | Check the card data |
| **96** | ICC Dynamic Data Length recovered is bigger than 8 | Check the card data |

# 9 Appendix

## A Return codes of the kernel and other macro definitions

### （1） Return codes of the kernel

| Return code | Value | Remark |
|---|---|---|
| EMV_OK | 0 | Operation succeeds. |
| ICC_RESET_ERR | -1 | IC card reset failed. |
| ICC_CMD_ERR | -2 | IC card command failed. |
| ICC_BLOCK | -3 | IC card has been blocked. |
| EMV_RSP_ERR | -4 | IC card response code error. |
| EMV_APP_BLOCK | -5 | The EMV application has been blocked. |
| EMV_NO_APP | -6 | There is no EMV application supported by terminal. |
| EMV_USER_CANCEL | -7 | Transaction is canceled by user. |
| EMV_TIME_OUT | -8 | User operation timeout. |
| EMV_DATA_ERR | -9 | IC card data format error. |
| EMV_NOT_ACCEPT | -10 | Transaction does not accept. |
| EMV_DENIAL | -11 | Transaction denied. |
| EMV_KEY_EXP | -12 | Key expired. |
| EMV_NO_PINPAD | -13 | There is no pinpad or pinpad does not work. |
| EMV_NO_PASSWORD | -14 | No PIN or user doesn’t want to input PIN. |
| EMV_SUM_ERR | -15 | Check sum error of key. |
| EMV_NOT_FOUND | -16 | The data element does not exist. |
| EMV_NO_DATA | -17 | No value for the data element. |

| | | |
|---|---|---|
| EMV_OVERFLOW | -18 | Memory overflow. |
| NO_TRANS_LOG | -19 | No transaction log. |
| RECORD_NOTEXIST | -20 | Specified log record does not exist. |
| LOGITEM_NOTEXIST | -21 | Specified item does not exist in the log. |
| ICC_RSP_6985 | -22 | ICC response with 6985 when GAC or GPO. |
| EMV_FILE_ERR | -24 | File error |
| EMV_PARAM_ERR | -30 | Parameter error. |

（2） Return codes or parameter of callback functions

| Return code | Value | Remark |
|---|---|---|
| REFER_APPROVE | 0x01 | Referral transaction approval. |
| REFER_DENIAL | 0x02 | Referral transaction denial. |
| ONLINE_APPROVE | 0x00 | Online transaction approval. |
| ONLINE_FAILED | 0x01 | Online transaction failed. |
| ONLINE_REFER | 0x02 | Online transaction referral. |
| ONLINE_DENIAL | 0x03 | Online transaction denial. |
| ONLINE_ABORT | 0x04 | Online transaction abort. |

（3） Other macro definitions

| Return code | Value | Remark |
|---|---|---|
| MAX_APP_NUM | 100 | Maximum number of application list. |
| MAX_KEY_NUM | 64 | Maximum number of public key stored in CA public key list of kernel for None Prolin platform. |
| | 7 | Maximum number of public key stored in CA public key list of kernel for Prolin platform. |
| MAX_REVOCLIST_NUM | 30 | Maximum number of revoked Issuer public key certification stored in kernel. |
| PART_MATCH | 0x00 | Application selection matching flag (partial matching). |
| FULL_MATCH | 0x01 | Application selection matching flag (full matching). |
| EMV_CASH | 0x01 | Transaction type (cash). |
| EMV_GOODS | 0x02 | Transaction type (goods). |
| EMV_SERVICE | 0x04 | Transaction type (service). |
| EMV_CASHBACK | 0x08 | Transaction type (cashback) |
| EMV_PED_TIMEOUT | 0x01 | PCI verify offline PIN, PIN input |

| | | |
|---|---|---|
| | | timeout.<br>Corresponding PED return code:<br>PED_RET_ERR_INPUT_TIMEOUT |
| EMV_PED_WAIT | 0x02 | PCI verify offline PIN, PIN input interval not enough.<br>Corresponding PED return code:<br>PED_RET_ERR_WAIT_INTERVAL |
| EMV_PED_FAIL | 0x03 | PCI verify offline PIN, other failure<br>Corresponding PED return code:<br>Other error codes |
| ATTR_N | 0x04 | n: numeric |
| ATTR_B | 0x08 | b: binary |
| ATTR_CN | 0x10 | cn: compressed numeric |
| ATTR_AN | 0x20 | an: alphabet and numeric |
| ATTR_ANS | 0x40 | ans: alphabet, numeric and special character |
| AC_AAC | 0x00 | The AC type is AAC |
| AC_TC | 0x01 | The AC type is TC |
| AC_ARQC | 0x02 | The AC type is ARQC |
| AC_AAC_HOST | 0x03 | The AC type is AAC since the online result is ONLINE_DENIAL, (only for VISA online) |

NOTE

All the function prototype and macros are defined in emvlib.h. So, if the application includes the emvlib.h, it can call all the functions and macros in the kernel.

## B EMV_PARAM

*EMV_PARAM*

```
typedef struct{
    unsigned char MerchName[256];
    unsigned char MerchCateCode[2];
    unsigned char MerchId[15];
    unsigned char TermId[8];
    unsigned char TerminalType;
    unsigned char Capability[3];
    unsigned char ExCapability[5];
    unsigned char TransCurrExp;
    unsigned char ReferCurrExp;
    unsigned char ReferCurrCode[2];
    unsigned char CountryCode[2];
    unsigned char TransCurrCode[2];
    unsigned long ReferCurrCon;
    unsigned char TransType;
    unsigned char ForceOnline;
   unsigned char GetDataPIN;
   unsigned char SurportPSESel;
}EMV_PARAM;
```

| Name | Description |
|---|---|
| **MerchName[256]** | merchant name<br>(usually no need to set) |
| **MerchCateCode[2]** | merchant catalog code<br>(usually no need to set) |
| **MerchInd[15]** | merchant identification<br>(should be set) |
| **TermInd[8]** | terminal identification<br>(should to be set) |
| **TerminalType** | terminal type |
| **Capability[3]** | terminal capability |
| **ExCapability[5]** | terminal extended capability |
| **CountryCode[2]** | terminal country code (default : “\x08\x40”)<br>USA : “\x08\x40” China : “\x01\x56”<br>Korea : “\x04\x10” Singapore : “\x07\x02” |
| **TransCurrCode[2]** | transaction currency code (default : “\x08\x40”)<br>USA: “\x08\x40” China: “\x01\x56”<br>Korea: “\x04\x10” Singapore: “\x07\x02” |
| **TransCurrExp** | transaction currency exponent (default : 0x02)<br>For example : RMB, USD, HKD: 0x02<br>Korean WON: 0x00 |
| **ReferCurrCode[2]** | reference currency code (default: “\x08\x40”) |
| **ReferCurrExp** | reference currency exponent (default: “0x02”) |
| **ReferCurrCon** | the conversion quotients between transaction currency and reference currency (default : 1000)<br>(the exchange rate of transaction currency to reference currency *1000) |
| **TransType** | set current transaction type<br>EMV_CASH or EMV_GOODS or EMV_SERVICE or EMV_GOODS& EMV_CASHBACK or EMV_SERVICE& EMV_CASHBACK<br>**(refer to appendix A for macro definitions)** |
| **ForceOnline** | merchant force online (1 means always online transaction) |

| | |
|---|---|
| **GetDataPIN** | read the IC card PIN retry counter before verify the PIN or not (1 : read, 0 : not read, default : 1) |
| **SurportPSESel** | support PSE selection mode or not<br>(1 : support, 0 : not support, default : 1) |

NOTE

After kernel initialized, the parameters above had been set by default value. The application can call the function EMVGetParameter(EMV_PARAM *tParam) to get the default value of the kernel, and modify some of them which need to or must be modified, and then store them in the kernel by calling EMVSetParameter(EMV_PARAM *tParam). The kernel will store the modified parameters in the file system, so they are still valid when the terminal power-on next time.

## C EMV_CAPK

***EMV_CAPK***

```
typedef struct{
        unsigned char RID[5];
        unsigned char KeyID;
        unsigned char HashInd;
        unsigned char ArithInd;
        unsigned char ModulLen;
        unsigned char Modul[248];
        unsigned char ExpLen;
        unsigned char Exp[3];
        unsigned char ExpDate[3];
        unsigned char CheckSum[20];
}EMV_CAPK;
```

| Name | Description |
|---|---|
| **RID[5]** | Registered Application Provider Identifier |
| **KeyID** | key index |
| **HashInd** | HASH arithmetic index |
| **ArithInd** | RSA arithmetic index |
| **ModulLen** | module length |
| **Modul[248]** | module |

| | |
|---|---|
| **ExpLen** | exponent length (1 or 3) |
| **Exp[3]** | exponent ("\x03" or "\x01\x00\x01") |
| **ExpDate[3]** | the expire date of the key (format : YYMMDD) |
| **CheckSum[20]** | key check sum |

## D EMV_APPLIST

***EMV_APPLIST***

```
typedef struct{
      unsigned char AppName[33];
      unsigned char AID[17];
      unsigned char AidLen;
       unsigned char SelFlag;
      unsigned char Priority;
      unsigned char TargetPer;
      unsigned char MaxTargetPer;
      unsigned char FloorLimitCheck;
      unsigned char RandTransSel;
      unsigned char VelocityCheck;
       unsigned long FloorLimit;
      unsigned long Threshold;
      unsigned char TACDenial[6];
      unsigned char TACOnline[6];
      unsigned char TACDefault[6];
       unsigned char AcquierId[6];
      unsigned char dDOL[256];
      unsigned char tDOL[256];
      unsigned char Version[3];
      unsigned char RiskManData[10];
}EMV_APPLIST;
```

| Name | Description |
|---|---|
| ***AppName[33]*** | *Local application name. The string ends with '\x00' and is 32 bytes in maximum.*<br>*Terminal can prompt for the application label or application preferred name of the IC card EMV application to let the cardholder to choose. But the language of the application label or application preferred name may be English and not convenient for the operator. In order to display the application name in local language, the kernel offers the setting of AppName.* |

| | |
|---|---|
| | *If AppName[0]=0, terminal will prompt for the application label or application prferred name.*<br>*If AppName is set with a application name, for example, "银联 EMV 卡"，the application name on the terminal will be prompted as "银联 EMV 卡".* |
| ***AID[17]*** | *Application ID, 16 bytes in maximum. (AID is corresponding to the AppName. The kernel searchs application according to the AID.)* |
| ***AidLen*** | *the length of AID* |
| ***SelFlag*** | *Application selection flag (partial matching PART_MATCH or full matching FULL_MATCH)*<br>*(**Refer to the macro definition in appendix A.**)* |
| ***Priority*** | *priority indicator*<br>*(It's returned by ICC, so nothing needs to be done by application.)* |
| ***TargetPer*** | *Target percent (0 – 99) (provided by acquirer)*<br>*(**Refer to the risk management in EMV specification.**)* |
| ***MaxTargetPer*** | *Max target percent(0 – 99) (provided by acquirer)*<br>*(Refer to the risk management in EMV specification.)* |
| ***FloorLimitCheck*** | *For the online only terminal , check the floor limit or not (1: check, 0 : not check，default:1)* |
| ***RandTransSel*** | *For the online only or offline only terminal , perform random transaction selection or not (1: perform, 0 : not perform, default : 1)* |
| ***VelocityCheck*** | *For the online only terminal , perform velocity check or not (1 : perform, 0 not perform, default : 1)* |
| ***FloorLimit*** | *Floor limit (provided by acquirer)*<br>*(**Refer to the risk management in EMV specification.**)* |
| ***Threshold*** | *Threshold (provided by acquirer)*<br>*(**Refer to the risk management in EMV specification**)*<br>*Notes: If TargetPer=99 and Threshold=0xffffffff, all the transaction will be done online.* |
| ***ActDenial[6]*** | *Terminal action code - denial*<br>*(default : "\x00\x10\x00\x00\x00" (Visa140))*<br>*(It must be set if acquirer provides it.)* |
| ***ActOnline[6]*** | *Terminal action code – online*<br>*(default : "\xD8\x40\x04\xF8\x00" (Visa140))*<br>*(It must be set if acquirer provides it.)* |
| ***ActDefault[6]*** | *Terminal action code – default*<br>*(default : "\xD8\x40\x00\xA8\x00" (Visa140))*<br>*(It must be set if acquirer provides it.)* |
| ***AcquierId[6]*** | *Acquirer identifier ( length between 6 to 11, compressed BCD format, pending 'F' on the right)*<br>*(It must be set if acquirer provides it, otherwise set 0x0.)* |
| ***dDOL[256]*** | *terminal default DDOL* |

| | |
|---|---|
| | *dDOL[0] is the length of DDOL, the others are the value of DDOL.*<br>*(default : "\x03\x9F\x37\x04" (Visa140))*<br>*(It must be set if acquirer provides it.)* |
| ***tDOL[256]*** | *terminal default TDOL*<br>*tDOL[0] is the length of TDOL, the others are the value of TDOL.*<br>*(default : "\x0F\x9F\x02\x06\x5F\x2A\x02\x9A\x03\x9C \x01\x95\x05\x9F\x37\x04" (Visa140))*<br>*(It must be set if acquirer provides it.)* |
| ***Version[3]*** | *application version*<br>*(It's returned by ICC, so nothing needs to be done by application)* |
| ***RiskManData[10]*** | *Risk management data*<br>*RiskManData[0] is the length of Risk management data, the others are the value of Risk management data.*<br>*(RiskManData[0] default : 0)*<br>*(It needn't be set unless issuer requires.)* |

## E EMV_REVOCLIST

***EMV_REVOCLIST***

```
typedef  struct
{
     unsigned  char    ucRid[5];
     unsigned  char    ucIndex;
     unsigned  char    ucCertSn[3];
}EMV_REVOCLIST;
```

| NAME | DESCRIPTION |
|---|---|
| **ucRid[5]** | Registered Application. |
| **ucIndex** | Certification Authenticate Public Key Index. |
| **ucCertSn** | Issuer Certificate Serial Number. |

## F ELEMENT_ATTR

***ELEMENT_ATTR***

```
typedef struct{
        int MaxLen;
        unsigned short Tag;
        unsigned short Attr;
        unsigned short usTemplate[2];
        unsigned char ucSource;
}ELEMENT_ATTR;
```

| Name | Description |
|---|---|
| **MaxLen** | The maximum length for this tag. |
| **Tag** | Tag |
| **Attr** | The format of this data<br>**#define ATTR_N 0x04 :numeric**<br>**#define ATTR_B 0x08 :binary**<br>**#define ATTR_CN 0x10 :compressed numeric**<br>**#define ATTR_AN 0x20 :alphabet and numric**<br>**#define ATTR_ANS 0x40 :alphabet, numeric and special character** |
| **usTemplate** | The template which this tag belongs, 0 if none |
| **ucSource** | The source of data element.<br>**#define EMV_SRC_TM 1 /* Terminal */**<br>**#define EMV_SRC_ICC 0 /* ICC */**<br>**#define EMV_SRC_ISS 2 /* Issuer */** |

## G APPLABEL_LIST

| *APPLABEL_LIST* |
|---|

```
typedef struct{
      unsigned char aucAppPreName[17];
      unsigned char aucAppLabel[17];
      unsigned char aucIssDiscrData[244];
      unsigned char aucAID[17];
      unsigned char ucAidLen;
}APPLABEL_LIST;
```

| Name | Description |
|---|---|
| **aucAppPreName[17]** | Application preferred name, ending with "\0" |
| **aucAppLabel[17]** | Application label, end with "\0" |
| **aucIssDiscrData[244]** | Data in template "BFOC" or "73", in the format of length+value, where 1 byte for length and other bytes for value |
| **aucAID[17]** | AID of ICC |
| **ucAidLen** | Length of AID of ICC |

## H EMV_EXTMPARAM

*EMV_EXTMPARAM*

```
typedef struct{
        unsigned char ucUseTermAIPFlg;
       unsigned char aucTermAIP[2];
      unsigned char ucBypassAllFlg;
} EMV_EXTMPARAM;
```

| Name | Description |
|---|---|
| **ucUseTermAIPFlg** | 0-TRM is based on AIP of card, 1-TRM is based on AIP of Terminal,  the default value is 0. |
| **aucTermAIP[2]** | The bit4 of byte1 decide whether force to perform TRM, “08 00 “- Yes; “00 00 “- No. Default value is “00 00”. |
| **ucBypassAllFlg** | Whether bypass all other pin when one pin has been bypassed 1-Yes, 0-No |

## I EMV_MCKPARAM

***EMV_MCKPARAM***

```
typedef struct{
     unsigned char   ucBypassPin;
    unsigned char   ucBatchCapture;
     void *pvoid;
}EMV_MCKPARAM;
```

| Name | Description |
|---|---|
| **ucBypassPin** | 0- Not supported，1-Supported. Default value is 1 |
| **ucBatchCapture** | 0-   ODC, 1-BDC. Default value is BDC<br>(ODC: Online Data Capture; BDC: Batch Data Capture) |
| **pvoid** | |

## J EMV_TMECPARAM

***APPLABEL_LIST***

```
typedef struct{
       unsigned char ucECTSIFlg;
      unsigned char ucECTSIVal;
     unsigned char ucECTTLFlg;
     unsigned long ulECTTLVal;
}EMV_TMECPARAM;
```

| Name | Description |
|---|---|
| **ucECTSIFlg** | TSI flag is exit or not.<br>TSI-Electronic cash terminal support indicator. |
| **ucECTSIVal** | TSI value. |
| **ucECTTLFlg** | TTL flag is exit or not. |

| | |
|---|---|
| | TTL-Electronic cash terminal transaction limit. |
| **ulECTTLVal** | TTLvalue |

## K APPLABEL_LIST

***APPLABEL_LIST***

```
typedef struct{
        unsigned char aucAppPreName[17];
        unsigned char aucAppLabel[17];
        unsigned char aucIssDiscrData[244];
        unsigned char aucAID[17];
        unsigned char ucAidLen;
}APPLABEL_LIST；
```

| Name | Description |
|---|---|
| **aucAppPreName[17]** | Application preferred name, ending with "\0" |
| **aucAppLabel[17]** | Application label, end with "\0" |
| **aucIssDiscrData[244]** | Data in template "BFOC" or "73", in the format of length+value, where 1 byte for length and other bytes for value |
| **aucAID[17]** | AID of ICC |
| **ucAidLen** | Length of AID of ICC |

## L Clss_TransParam

***Clss_TransParam***

```
typedef struct{
         unsigned long   ulAmntAuth;
         unsigned long   ulAmntOther;
         unsigned long   ulTransNo;
        unsigned char   ucTransType;
        unsigned char   aucTransDate[4];
        unsigned char   aucTransTime[4];
}Clss_TransParam;
```

| NAME | DESCRIPTION |
|---|---|
| **ulAmntAuth** | Authorize amount(unsigned long), for cash back, the amount required to include the amount of ulAmntOther. |
| **ulAmntOther** | Other amount(unsigned long) |
| **ulTransNo** | Transaction Sequence Counter(4 BYTE) |
| **ucTransType** | Transaction type'9C', 0-Consumer/Services 1-Cash/Cash back |
| **aucTransDate[4]** | Transaction Date YYMMDD |
| **aucTransTime[4]** | Transaction time HHMMSS |

## M RSA_PINKEY

*RSA_PINKEY*

```
typedef struct
{
        unsigned int    modlen;
        unsigned char mod[256];
        unsigned char exp[4];
        unsigned char iccrandomlen;
        unsigned char iccrandom[8];
}RSA_PINKEY;
```

| NAME | DESCRIPTION |
|---|---|
| **modlen** | PIN encryption key modulus. |
| **mod[256]** | PIN encryption key modulus, high byte first, low byte after, insufficient bit complement 0. |
| **exp[4]** | PIN encryption public key index, high byte first, low byte after, insufficient bit complement 0. |
| **iccrandomlen** | The random number length obtained from card. |
| **iccrandom[8]** | The random number obtained from card. |

## N APDU_SEND

***APDU_SEND***

```
        typedef struct
        {
                uchar           Command[4];
                ushort          Lc;
                uchar           DataIn[512];
                ushort          Le;
}APDU_SEND;
```

| NAME | DESCRIPTION |
|---|---|
| **Command[4]** | Command[] = {CLA, INS, P1, P2}. |
| **Lc** | The length of DataIn. |
| **DataIn[512]** | The data pointer which is sent to IC card. |
| **Le** | The expect length receive from IC card. The actual length is related to the specific command, which can get from the parameter LenOut in response structure APDU_RESP. |

## O APDU_RESP

***APDU_RESP***

```
typedef struct
{
        ushort          LenOut;
        uchar           DataOut[512];
        uchar           SWA;
        uchar           SWB;
}APDU_RESP;
```

| NAME | DESCRIPTION |
|---|---|
| **LenOut** | The actual length receives from IC card. |
| **DataOut[512]** | The pointer receives from IC card. |
| **SWA** | Status word A. |
| **SWB** | Status word B. |

# PAX EMV Kernel API Programming Guide

PAX Technology Limited

www.pax.com.hk

Hong Kong
Room 2416, 24/F, Sun Hung Kai Centre, 30 Harbour Road,
Wanchai, Hong Kong
Tel: +852-25888800
Fax: +852-28023300

Shenzhen
4/F, No.3 Building, Software Park, Second Central Science-Tech Road,
High-Tech Industrial Park, Shenzhen, Guangdong 518057, P.R. China
Tel: +86-755-86169630
Fax: +86-755-86169634