SmartCard L2 BASE API

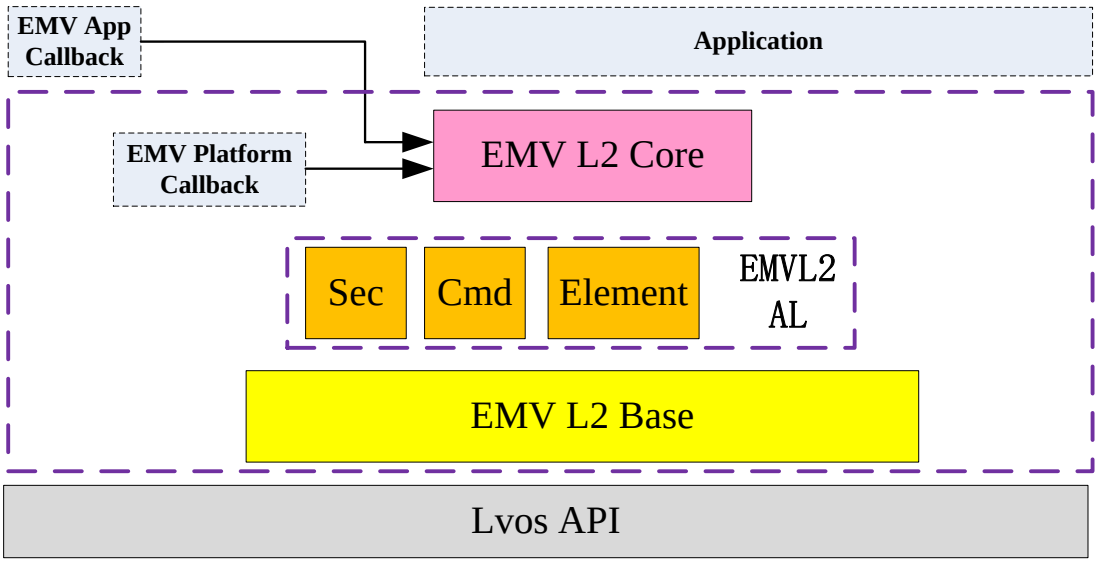
2022-11-09

Revision History

No	Ver	Fix date	Name	description
1	V1.0	2022-11-08	xiaoty	new

directory

1 EMV Framework



2 Emv L2 Core API

2.1 EMVL2CoreInit

Prototype	int EMVL2CoreInit(void)	
Function	kernel initialization	
Parameter	Input	
	Output	
Return	EMV_RET_OK, Other error Codes	
Notice	Call once on startup	

2.2 EMVL2GetKernelType

Prototype	int EMVL2GetKernelType(void)	
Function	get the value of kernal type	
Parameter	Input	
	Output	
Return	The type value of the kernel—refer to 3.6	
Notice		

2.3 EMVL2PreProcess

Prototype	int EMVL2PreProcess(void)	
Function	contactless transaction preprocessing	
Parameter	Input	
	Output	
Return	EMV_RET_OK, Other error Codes	
Notice	Only for contactless transaction	

2.4 EmvL2AppSel

Prototype	int EMVL2AppSel(u32 Cardtype, u32 *CandListNum, ST_CANDLIST *CandList)	
Function	application selection	
Parameter	Input	Cardtype: 1:contactless 2:contact
	Ouput	CandListNum:the num of applist CandList:the list of app info(refer to 3.1)
Return	EMV_RET_OK, Other error Codes	
Notice		

2.5 EMVL2AppFinalSel

Prototype	int EMVL2AppFinalSel(u32 FinalNo)	
Function	application final selection	
Parameter	Input	FinalNo: the index in the CandList
	Ouput	
Return	EMV_RET_OK, Other error Codes	
Notice		

2.6 EmvGetCandListBySeqNumber

Prototype	int EmvGetCandListBySeqNumber(int no, ST_CANDLIST * finalApp)	
Function	application final selection	
Parameter	Input	no: the index in the CandList
	Ouput	finalApp:
Return	EMV_RET_OK, Other error Codes	
Notice	contactless transaction excute or if app select return CandListNum is 0,this function can excute.	

2.7 EMVL2InitApp

Prototype	int EMVL2InitApp(void)	
Function	application initialization	
Parameter	Input	
	Ouput	
Return	EMV_RET_OK, Other error Codes	
Notice		

2.8 EMVL2ReadAppData

Prototype	int EMVL2ReadAppData(void)	
Function	Read application data	
Parameter	Input	
	Ouput	
Return	EMV_RET_OK, Other error Codes	
Notice		

2.9 EMVL2DataAuth

Prototype	int EMVL2DataAuth(void)	
Function	application data authentication	
Parameter	Input	
	Ouput	
Return	EMV_RET_OK, Other error Codes	
Notice		

2.10 EMVL2ProcRestric

Prototype	int EMVL2ProcRestric(void)	
Function	limitations of processing	
Parameter	Input	
	Ouput	
Return	EMV_RET_OK, Other error Codes	
Notice		

2.11 EMVL2TermRiskManage

Prototype	int EMVL2TermRiskManage(void)

Function	terminal risk management	
Parameter	Input	
	Ouput	
Return	EMV_RET_OK, Other error Codes	
Notice		

2.12 EMVL2CVMProcess

Prototype	int EMVL2CVMProcess(u32 CVMStep, u32 CVMProcResult,u8 *CVMType)	
Function	Card holder authentication	
Parameter	Input	CVMStep: 0 / CVM_STEP_NEXT CVMProcResult: refer to 3.2
	Ouput	CVMType: see to notice
Return	CVM_STEP_NEXT(5), other codes finished	
Notice	#define CVM_FAIL_CVM 0x00 // fail CVM #define CVM_PLAIN_PIN 0x01 // plaintext PIN #define CVM_ONLINE_PIN 0x02 // online enciphered PIN #define CVM_PPIN_SIG 0x03 // plaintext PIN + signature #define CVM_ENCIPH_PIN 0x04 // enciphered PIN #define CVM_EPIN_SIG 0x05 // enciphered PIN + signature #define CVM_SIG 0x1E // signature #define CVM_NO_CVM 0x1F // no CVM #define CVM_CONSUMER_DEVICE 0x21 // no CVM #define CVM_CERT 0x20 // cardholder certificate #define CVM_FAIL_NEXT 0x40 // Apply succeeding CV Rule if this CVM is unsuccessful #define CVM_NULL 0xFF	

2.13 EMVL2TermActAnalysis

Prototype	int EMVL2TermActAnalysis(void)	
Function	terminal behavior analysis	
Parameter	Input	
	Ouput	
Return	EMV_RET_OK, Other error Codes	
Notice	the result detail refer to 3.7 CONTINUE/ APPROVE/ ONLINE_REQUEST/ DECLINED	

2.14 EMVL2Completion

Prototype	int EMVL2Completion(u32 OnlineResult, u32 Scriptlen, u8 *Script, u32 Issuerlen, u8 *IssuerData)	
Function	trsaction completed	
Parameter	Input	OnlineResult:refer to 3.3 Scriptlen:the length of script Script: Issuerlen:the length of issue data IssuerData:the issue data
	Ouput	
Return	APPROVE(1), Other error finished	
Notice		

2.15 EMVL2BaseTLVOperate

Prototype	int EMVL2BaseTLVOperate(u32 OpCode, u32 TagListLen,u8 *TagList, u32 *TLVDataLen,u8 *TLVData)	
Function	operating kernel data	
Parameter	Input	OpCode：refer to 3.4 TagListLen： TagList：the tag of query condition
	Ouput	TLVDataLen： TLVData：
Return	EMV_RET_OK, Other error Codes	
Notice		

2.16 EMVL2BaseParamOperate

Prototype	int EMVL2BaseParamOperate(u32 OpCode, ST_PARAM_EXTEND * ExParam)	
Function	operating kernel param	
Parameter	Input	OpCode: detail refer to 3.4 ExParam :tlv struct detail refer to 3.4
	Input/ Ouput	ExParam:refer to 3.5
Return	EMV_RET_OK, Other error Codes	
Notice		

2.17 EMVL2PINVerify

Prototype	int EMVL2PINVerify (void)	
Function	offline pin verify	
Parameter	Input	
	Ouput	
Return	EMV_REENTER_PIN(6), EMV_REENTER_PIN_LAST(7),other code continue	
Notice		

3 Definition of structure

3.1 CANDLIST

```
typedef struct
{
    u16 Index;          // The ordinal number applied to the candidate column
    u8   AID[16];               // AID，end of '\0'
    u8   AidLen;                // AID len
    u8   AppPreName[17];        // the preferred application name，end of '\0'
    u8   AppLabel[17];          //application label，end of '\0'
    u8   IssDiscrData[244];     //tag 'BF0C'：1byte+'BF0C' max to 222 bytes，end of '\0'
     u8   Priority;             // Priority flag
    u8   AppName[33];           // Local Application Name，end of '\0'
    u8   KernType;              //contactless kernel type
} ST_CANDLIST; /*align 4 Bytes*/
```

3.2 CVM_R

```
typedef enum
{
    CVM_SUCSEC,
    CVM_PINCANCEL,
    CVM_PINBYPASS,
    CVM_PINPADERROR,
    CVM_CERT_FAIL,
}CVM_R;
```

3.3 EMV_ONLINE_OUTCOME

```
typedef enum {
	ONLINE_APPROVE,
	ONLINE_DENIAL,
	ONLINE_FAILED,
	ONLINE_DENIAL_05
}EMV_ONLINE_OUTCOME;
```

3.4 EMV_STORE_OP

```
typedef enum
{
	OP_UPDATE = 1,
	OP_CLEAR,
	OP_DEL,
	OP_QUERY,
	OP_ENUM,
	OP_INITALL,
	OP_UPDATE_PAYPASS,
	OP_SET_NOT_PRESENT,
	OP_UPDATE_PAYWAVE,
	OP_QUERY_PAYWAVE,
	OP_QUERY_PAYPASS,
}EMV_STORE_OP;

typedef struct
{
	u32	tag;	//the value refer to EMV_PARAM_TYPE
	u32	len;	//sizeof the struct tag data
	void* value;	//the addr of the struct tag data
}ST_PARAM_EXTEND;

typedef enum
{
	TYPE_ALL = 0,
	TYPE_CAPK,
	TYPE_REVOCLIST,
	TYPE_CANDLIST,
	TYPE_TERMAIDLIST,
	TYPE_TERMPARAM,
	TYPE_PREPARAM,
	TYPE_TRANSOUTCOME,
```

```
    TYPE_AIDPARAM,
} EMV_PARAM_TYPE;
```

3.5 ST_TERMPARAM

```
typedef struct{
    u8   IFDSn[8];                  // IFD serial no 9F1E
    u8   TerminalType;              // terminal type 9F35
    u8   CountryCode[2];            // Terminal country code 9F1A
    u8   ForceOnline;               // Merchant compulsory on-line(1 always online)
    u8   GetDataPIN;                // Number of read retries before password detection
    u8   SurportPSESel;             // Whether to support the PSE option
    u8   UseTermAIPFlg;             // Whether to conduct risk management based on card AIP,0-card AIP,1- terminal AIP,default 0
    u8   TermAIP[2];                // Whether the terminal forcibly performs risk management，byte1-bit4 为 1：force；byte1-bit4 为 0：no 。default Both bytes are 0。
    u8   BypassAllFlg;              // whether bypass all other pin when one pin has been bypassed 1-Yes, 0-No
    u8   BypassPin;                 // 0-not surport，1—surport，default surport
    u8   BatchCapture;              // 0-online data capture, 1-batch capture
    u8   ECTSIFlg;                  // TSI exist? 1-exist （EC Terminal Support Indicator）
    u8   ECTSIVal;                  // Electronic cash terminal support indicator = 1,支持
    u8   ECTTLFlg;                  // TTL exist? 1-exist（EC Terminal Transaction Limit）
    u8   ECTTLVal[6];               // EC Terminal Transaction Limit
    u8   Capability[3];             // Terminal performance 9F33
    u8   AddCapability[5];          // Terminal expansion performance 9F40
    u8   ScriptMode;
    u8   AdviceFlag;
    u8   IsSupportSM;               //
    u8   IsSupportTransLog;         // Whether to support trading LOG
    u8   IsSupportMultiLang;        // Whether multiple languages are supported
    u8   IsSupportExceptFile;       // Whether exception files are supported
    u8   IsSupportAccountSelect; // Whether to support account selection
    u8   TTQ[4];                    // Terminal transaction attributes (ctls used)
    u8   IsReadLogInCard;           // Whether to read the card transaction record application selection process
    u8   reserved[3];                 //must be 0
}ST_TERMPARAM; /*align 4 Bytes*/
```

3.6 EMV_KERNEL_TYPE

```
typedef enum
{
```

```
    EMV = 0,
    QPBOC = 1,
    PAYPASS = 2,
    PAYWAVE = 3,
    AMERICAEXPRESS = 4,
    DISCOVER = 5,
    JCB = 6,
    EMV_KERNEL_MAX
}EMV_KERNEL_TYPE;
```

3.7 the kernel result value

```
#define SELECT_NEXTAPP_MAXLIMIT_EXCEED      8
#define EMV_REENTER_PIN_LAST          7
#define EMV_REENTER_PIN               6
#define CVM_STEP_NEXT                 5
#define TRY_AGAIN                     4
#define SELECT_NEXT_APP               3
#define ONLINE_REQUEST                2
#define APPROVE                       1
#define CONTINUE                      0
#define DECLINED                                -4000
#define TRY_ANOTHER_INTERFACE                    -4001
#define ENDAPPLICATION                          -4002
#define SEE_PHONE                                -4003
#define DECLINED_CAPKINREVO                      -4004
#define ONLINE_REQUEST_CAPKINREVO                -4005
#define FINALSELECT_DATA_ERR                    -4006
#define ENDAPPLICATION_EXCEPTFILE               -4007
#define ENDAPPLICATION_OTHERCARD                 -4008
#define SEEPHONE_CMD_SWAB_6986 -4099
#define ENDAPPLICATION_CMD_ERR                   -4100
#define ENDAPPLICATION_CMD_TIMEOUT               -4101
#define ENDAPPLICATION_CMD_SWAB_6985             -4102
#define ENDAPPLICATION_CMD_RSP_ERR               -4103
#define ENDAPPLICATION_CARD_BLOCK                -4104
#define ENDAPPLICATION_APP_BLOCK                 -4105
#define ENDAPPLICATION_TMAPP_EMPTY               -4106
#define ENDAPPLICATION_NO_SCAPP                  -4107
#define ENDAPPLICATION_DATA_ERR                  -4108
#define ENDAPPLICATION_DATA_DUPLICATE           -4109
#define ENDAPPLICATION_NOT_ACCEPT                -4110
#define ENDAPPLICATION_CARD_EXPIRED              -4111
#define EMV_NO_PREPARAM                          -4112
```

```
#define ENDAPPLICATION_L1_TIMEOUT_ERR                -4113
#define ENDAPPLICATION_L1_TRANSMISSION_ERR           -4114
#define ENDAPPLICATION_L1_PROTOCAL_ERR               -4115
#define ENDAPPLICATION_L2_CARD_DATA_MISSING          -4116
#define ENDAPPLICATION_L2_CAM_FAIL                   -4117
#define ENDAPPLICATION_L2_STATUS_BYTE                -4118
#define ENDAPPLICATION_L2_PARSING_ERR                -4119
#define ENDAPPLICATION_L2_MAX_LIMIT_EXEED            -4120
#define ENDAPPLICATION_L2_CARD_DATA_ERR              -4121
#define ENDAPPLICATION_L2_MAG_NOT_SUPPORT            -4122
#define ENDAPPLICATION_L2_NO_PPSE                    -4123
#define ENDAPPLICATION_L2_PPSE_FAULT                 -4124
#define ENDAPPLICATION_L2_EMPTY_CAND_LIST            -4125
#define ENDAPPLICATION_L2_IDS_READ_ERR               -4126
#define ENDAPPLICATION_L2_IDS_WRITE_ERR              -4127
#define ENDAPPLICATION_L2_IDS_DATA_ERRR              -4128
#define ENDAPPLICATION_L2_IDS_NO_MATCH_AC            -4129
#define ENDAPPLICATION_L2_TERMINAL_DATA_ERR          -4130
#define ENDAPPLICATION_L3_TIMEOUT                    -4131
#define ENDAPPLICATION_L3_STOP                       -4132
#define ENDAPPLICATION_L3_AMOUNT_NOT_PRESENT         -4133
#define ENDAPPLICATION_REPRESENT_CARD                -4134
#define ENDAPPLICATION_OHTER_CARD_WITHRECORD         -4135
#define ENDAPPLICATION_OHTER_CARD                    -4136
#define ENDAPPLICATION_CMD_RSP_ERR_GPO               -4137
#define ENDAPPLICATION_L2_CARD_DATA_FINALSEL         -4138
#define ENDAPPLICATION_L3_NO_DET_DATA                -4139
#define ENDAPPLICATION_KERNEL_NOT_SUPPORT            -4140
#define ENDAPPLICATION_CLSS_LIMIT_EXCEED             -4141
#define ENDAPPLICATION_ZERO_AMOUNT                   -4142
#define TRY_ANOTHER_INTERFACE_PREPROC                -4144
#define EMV_INVALID_PARAM                            -4500
#define EMV_SUM_ERR                                  -4501
#define EMV_PARAM_NOT_EXIST                          -4502
#define EMV_PARAM_DATA_ERROR                         -4503
#define PBOC_NO_LOG                                  -4504
#define PBOC_LOG_DATA_ERR                            -4505
#define EMV_NO_DATA -4506
#define PBOC_NO_LOG_FMT -4507 // 20180731
```

4 Examples of Usage

4.1 Module initialization

Start of each application，the following function should be called to initialize this module:

- EMVL2CoreInit

4.2 Parameter download

All parameter files must be stored in the files saved by the application. The kernel does not store files. The terminal achieves parameter storage and loading to the kernel through the EMV control library of the application. The control library interfaces are applied in the following order：

- EMV_SetDefault to set default terminal parameters
- EMV_AddAID function to set the AID list parameters, parameter structure refer to ST_AID. The aid list needs to be downloaded one by one.
- EMV_AddCAPK function to set the public key parameters, parameter structure refer to ST_CAPK. The public key needs to be downloaded one by one.

The default terminal parameters and the list of aid and capk parameters refer to the emvproc.c code. The application can edit the default parameters or load them from the platform as required.

4.3 EMV Process Implementation

Take SALE transaction as an example to illustrate the implementation method of EMV transaction, where EMV_EMVProcess_Simple and other functions refer to the emv.c template:

4.3.1 /* If the ic card is powered on, the following emv process is started*/
4.3.2 /* Processing flow before EMV terminal behavior analysis*/

```
int EMV_EMVProcess()
{
    /* emv simple process, application selection and final selection and reading card information*/
    iRet = EMV_EMVProcess_Simple();
    /* Card number confirmation callback*/
    iRet = EMV_Callback(1, NULL);
    /*Load the capk parameter to the kernel*/
    iRet = EMV_LoadCAPK2Kernel();
```

```
    /* Offline data authentication*/
    iRet = EMVL2DataAuth();
    /* Processing limitation*/
    iRet = EMVL2ProcRestric();
    /* Card holder authentication*/
    iRet = EMV_CVMProcess(&cvmtype);
    /* Terminal risk management*/
    iRet = EMVL2TermRiskManage();
    /* Terminal behavior analysis*/
    iRet = EMVL2TermActAnalysis();
}
```

4.3.3 EMV complete transaction process

```
int EMV_Process(int cardType)
{
    iRet = EMV_EMVProcess();
    /* According to the terminal behavior analysis result, the corresponding processing is carried out*/
    if(iRet == CONTINUE || iRet == ONLINE_REQUEST || iRet == APPROVE) {
        EMV_ReadTransData();
    }
    /*On-line processing, do the second GAC, script processing*/
    if(iRet == ONLINE_REQUEST) {
        iRet = EMV_OnlineProcess();
        return iRet;
    }

    switch(iRet)
    {
      case APPROVE:
      case DECLINED:
      case TRY_AGAIN:
      case TRY_ANOTHER_INTERFACE:
      case TRY_ANOTHER_INTERFACE_PREPROC:
      case EMV_RET_USER_ABORT:
      default:
    }
}
```

4.4 Transaction flow