EMV Contactless API for Vega3000

Reference Manual

Version 0.95

August 2014

Doc.#

Confidential Level:

Table of Contents

Revision History

Version	Date	Editor	Description
V0.9	2014.03.24	Weber Ku	Release.
V0.91	2014.07.05	Weber Ku	1. Modify the Tag 9F53 in transaction related data of EMVCL_StartTransactionEx/EMVCL_InitTransactionEx for Visa to "Not Supported" 2. Typo fixed 3. Add EMVCL_SpecialEventRegister API 4. Add Special Event description
V0.92	2014.07.24	Weber Ku	Add Forced Transaction Online information in Transaction Releated Data.
V0.93	2014.07.31	Weber Ku	Add EMVCL_CompleteEx API
V0.94	2014.08.06	Weber Ku	1. Add Application Prefer Order information in Transaction Releated Data. 2. Add Transaction Functions Notes
V0.95	2014.08.13	Weber Ku	1. Add EMVCL_DETECT_TXN_DATA structure 2. Add EMVCL_DetectTransactionEx API

WARNING

1 Introduction

This document defines an interface to communicate with EMV contactless approved kernel for performing a contactless transaction on the pinpad/terminal.

2 Application Interface

2.1 DATA STRUCTURES

2.1.1 EMVCL_EVENT

```
typedef struct
{
   BOOL (*OnCancelTransaction)(void);
   void (*OnShowMessage)(IN BYTE bKernel, IN
          EMVCL_USER_INTERFACE_REQ_DATA
          *baUserInterfaceRequestData);

}EMVCL_EVENT;
```

PARAMETER(S)	
OnCancelTransaction	(1) Implement a function for cancel transaction. When the event occurs, returning TRUE indicates that the current transaction will be canceled, while returning FALSE indicates to continue the processing. (2) If OnCancelTransaction is not implemented (i.e Event_fCancelTransaction is set to NULL). No cancellation will be performed.
OnShowMessage	(1) Implement a function to show messages during transaction. bKernel: Indicates which kernel to request this message to display

	baUserInterfaceRequestData: the information from kernel to inform customer about the progress of transaction. (Please refer to "EMV Contactless Specification for Payment Systems Book A" for detail) (2) If OnShowMessage is not implemented. No message will be shown during transaction.

2.1.2 EMVCL_INIT_DATA

```
typedef struct
{
        BYTE bConfigFilenameLen;
        BYTE *pConfigFilename;
        EMVCL_EVENT stOnEvent;

}EMVCL_INIT_DATA;
```

NOTE

The config file is a setting file which includes Tags, CAPK, Parameters, etc.
If pConfigFilename is not NULL and bConfigFilenameLen is not equal to 0, the setting of the config file will be set to contactless kernels.
If bConfigFilenameLen is equal to 0, no setting will be done.

PARAMETER(S)	
bConfigFilenameLen	The length of the config file name.
pConfigFilename	Config file name
stOnEvent	Point to struct EMVCL_EVENT

2.1.3 EMVCL_USER_INTERFACE_REQ_DATA

```
typedef struct
{
      BYTE    bMessageIdentifier;
      BYTE    bStatus;
      BYTE    baHoldTime[3];
      BYTE    baLanguagePreference[8];
      BYTE    bValueQualifier;
      BYTE    baValue[6];
```

```
        BYTE     baCurrencyCode[2];

    }EMVCL_USER_INTERFACE_REQ_DATA;
```

NOTE	
Please refer ro User Req Data Informatin below and EMV Contactles Specification for Payment Systems Book A for more detail.	
PARAMETER(S)	
bMessageIdentifier	Indicates the text string to be displayed
bStatus	Identifies the status of the transaction (for example when the card can be removed)
baHoldTime	If provided, indicates that the reader is to delay the processing of the Message Identifier in the next User Interface Request until the Hold Time has elapsed. The Hold Time is an integer in units of 100ms; the default value is zero.
baLanguagePreference	If Language Preference (as per EMV Tag '5F2D' according to ISO 639-1) is present and if the language is supported by the reader or terminal, then the message identified and all subsequent messages to the cardholder are displayed in this language until the transaction concludes.
bValueQualifier	Value Qualifier is defined as either “Amount” or “Balance”.
baValue	If Value Qualifier is present and equals “Amount” or “Balance”, the Value will be displayed as an amount or balance in conjunction with the message. The representation will be according to the Currency Code with leading zeros suppressed whilst leaving at least one significant digit (so that an amount of 0.09 would be shown as 0.09 and not as .09, 09, or 9).
baCurrencyCode	Numeric value as per ISO 4217

2.1.4 EMVCL_ACT_DATA

```
    typedef struct
    {
        IN BYTE        bStart;
        IN BYTE        bTagNum;
        IN USHORT      usTransactionDataLen;
        IN BYTE        *pbaTransactionData;

    }EMVCL_ACT_DATA;
```

PARAMETER(S)	
bStart	Indicate the status of start transaction.
= 0x00 (Start A)	Start a new transaction
= 0x01 (Start B)	Restart the transaction. Start B may be used by below: (1)Outcome is Try again (2)Transaction is terminated but kernel request restart(See phone, second tap…)
bTagNum	Number of Transaction related Data
usTransactionDataLen	Transaction related Data Length.
pbaTransactionData	Transaction related Data. (format is TLV1 + TLV2 +..+ TLVn)

2.1.5 EMVCL_RC_DATA_EX

```
typedef struct
{
    OUT BYTE       bSID;
    OUT BYTE       baDateTime[15];
    OUT BYTE       bTrack1Len;
    OUT BYTE       baTrack1Data[100];
    OUT BYTE       bTrack2Len;
    OUT BYTE       baTrack2Data[100];
    OUT USHORT     usChipDataLen;
    OUT BYTE       baChipData[1024];
    OUT USHORT     usAdditionalDataLen;
    OUT BYTE       baAdditionalData[1024];
}EMVCL_RC_DATA_EX;
```

PARAMETER(S)	
bSID	Scheme Identifier
= 0x13	VISA Old US
= 0x16	VISA Wave 2

= 0x17	VISA qVSDC
= 0x18	VISA MSD
= 0x20	PayPass MagStripe
= 0x21	PayPass MChip
= 0x61	JCB Wave 2
= 0x62	JCB qVSDC
= 0x50	AMEX EMV
= 0x52	AMEX MagStripe
= 0x41	DISCOVE Zip
baDateTime	YYYYMMDDHHMMSS format
bTrack1Len	Track 1 data length
baTrack1Data	Track 1 data It's ANS format. (Variable.up to 76)
bTrack2Len	Track 2 data length
baTrack2Data	Track 2 data It's ASCII format (Variable.up to 19)
usChipDataLen	Chip data length (=0:　Chip data is not present)
baChipData	Chip data.
usAdditionalDataLen	Additional data length (=0: additional is not present)
baAdditionalData	Additional data.

2.1.6 EMVCL_RC_DATA_ANALYZE

```
typedef struct
{
   OUT USHORT  usTransResult;
   OUT BYTE       bCVMAnalysis;
   OUT BYTE       baCVMResults[3];
   OUT BYTE       bVisaAOSAPresent;
   OUT BYTE       baVisaAOSA[6];
   OUT BYTE       bODAFail;

}EMVCL_RC_DATA_ANALYZE;
```

PARAMETER(S)	
usTransResult	Indicate transaction result
= 0x0002	Approval
= 0x0003	Decline
= 0x0004	Online
bCVMAnalysis	Indicate which CVM is required
= 0x00	None

= 0x01	Signature
= 0x02	Online PIN
= 0x03	CVM Fail
= 0x04	NO CVM
= 0x05	Confirmation Code Verified
baCVMResults	CVM Result value
bVisaAOSAPresent	Indicate if Visa Available Offline Spending Amount(AOSA) is present
= 0x00	NOT Present
= 0x01	Present
baVisaAOSA	AOSA Value
bODAFail	Indicate if Offline Data Authentication is fail
= 0x00	ODA OK
= 0x01	ODA Fail

2.1.7 EMVCL_AID_SET_TAG_DATA

```
typedef struct
{
   IN BYTE        bAIDLen;
   IN BYTE        baAID[16];
   IN BYTE        bKernelID;
   IN BYTE        bTransactionType;
   IN USHORT      usTAGDataLen;
   IN BYTE        *pbaTAGData;

}EMVCL_AID_SET_TAG_DATA;
```

NOTE
(1) Set different tags setting for different combination list {TransType – AID-KERNEL ID}
(2) bTransactionType, baAID, bKernelID are made a combination list and this combination list maintain a tags setting which is save via pbaTAGData.

PARAMETER(S)	
bAIDLen	AID Length
baAID	AID
bKernelID	Kernel ID
= 0x02	MasterCard PayPass 3.x

= 0x03	Visa VCPS 2.1.x
bTransactionType	Transaction Type
usTAGDataLen	Tags Setting Length
pbaTAGData	Point to Tags Setting which will be saved into EMVCL kernel

2.1.8 EMVCL_AID_GET_TAG_DATA

```
typedef struct
{
   IN BYTE           bAIDLen;
   IN BYTE           baAID[16];
   IN BYTE           bKernelID;
   IN BYTE           bTransactionType;
   INOUT USHORT      usTAGDataLen;
   OUT BYTE          *pbaTAGData;
}EMVCL_AID_GET_TAG_DATA;
```

PARAMETER(S)	
bAIDLen	AID Length
baAID	AID
bKernelID	Kernel ID
bTransactionType	Transaction Type
usTAGDataLen	Tags Setting Length
pbaTAGData	Point to Tags Setting which will be got from EMVCL kernel

2.1.9 EMVCL_CA_PUBLIC_KEY

```
typedef struct
{
   BYTE        bAction;
   BYTE        bIndex;
   UINT        uiModulusLen;
   BYTE        baModulus[248];
   UINT        uiExponentLen;        // Length of Extension
   BYTE        baExponent[3];        // Extension
   BYTE        baHash[20];           // Key Hash (SHA-1) Result
}EMVCL_CA_PUBLIC_KEY;
```

PARAMETER(S)	
bAction	
= 0x00 Add	Add one CAPK
= 0x01 Delete	Delect one CAPK
= 0x02 Delect All	Delete All CAPK
bIndex	CAPK Index
uiModulusLen	Modulus Length
baModulus	Modulus
uiExponentLen	Exponent Length
baExponent	Exponent
baHash	The hash is calculated on the concatenation of RID, Key Index, Modulus, and Exponent using SHA-1 algorithm

2.1.10 EMVCL_PARAMETER_DATA

```
typedef struct
{
   UINT          uiNop;
   UINT          uiIndex[100];
   UINT          uiLen[100];
   BYTE          baData[100][20];
}EMVCL_PARAMETER_DATA;
```

PARAMETER(S)	
uiNop	Number of Parameter
uiIndex	Parameter Index
uiLen	Length of Parameter Data
baData	Parameter Data

2.1.11 EMVCL_SCHEME_DATA

```
typedef struct
{
   BYTE          bNoS;
   BYTE          baID[255];
```

```
    BYTE          baAction[255];
}EMVCL_SCHEME_DATA;
```

PARAMETER(S)	
bNoS	Number of Scheme
baID	Scheme ID
baAction	Active or deactive

2.1.12 EMVCL_DETECT_TXN_DATA

```
typedef struct
{
    IN BYTE         bTagNum;
    IN USHORT       usTransactionDataLen;
    IN BYTE         *pbaTransactionData;
    INOUT BYTE      bSelectedAIDLen;
    OUT BYTE        *pbaSelectedAID;
    INOUT ULONG     ulSelectAIDRspLen;
    OUT BYTE        *pbaSelectAIDRsp;
    INOUT ULONG     ulSelectPPSERspLen;
    OUT BYTE        *pbaSelectPPSERsp;
}EMVCL_DETECT_TXN_DATA;
```

PARAMETER(S)	
bTagNum	Number of Transaction related Data
usTransactionDataLen	Transaction related Data Length
pbaTransactionData	Transaction related Data. (format is TLV1 + TLV2 +..+ TLVn)
bSelectedAIDLen	[IN] pbaSelectedAID buffer size [OUT] Selected AID length
pbaSelectedAID	Point to a buffer to get AID.
ulSelectAIDRspLen	[IN] pbaSelectAIDRsp buffer size [OUT] Select AID Response length
pbaSelectAIDRsp	Point to a buffer to get the response of Select AID
ulSelectPPSERspLen	[IN] pbaSelectPPSERsp buffer size [OUT] Select PPSE Response length
pbaSelectPPSERsp	Point to a buffer to get the response of Select PPSE

2.2 Gerneral Functions

2.2.1 EMVCL_Initialize

Initiate EMVCL environment.

PROTOTYPE	
ULONG	**EMVCL_Initialize(EMVCL_INIT_DATA *pInitData);**
PARAMETER(S)	
[IN] pInitData	Point to struct EMVCL_INIT_DATA
RETURN VALUE	
d_EMVCL_NO_ERROR d_EMVCL_INIT_TAGSETTING_ERROR d_EMVCL_INIT_CAPK_ERROR d_EMVCL_INIT_PARAMETER_ERROR d_EMVCL_INIT_CAPABILITY d_EMVCL_INIT_REVOCATION_ERROR d_EMVCL_INIT_EXCEPTION_FILE_ERROR	
REMARK	
NONE	
EXAMPLE(S)	

```
ULONG ulRtn;
EMVCL_INIT_DATA emvcl_initdat;

emvcl_initdat.stOnEvent.OnCancelTransaction = NULL;
emvcl_initdat.stOnEvent.OnShowMessage = ShowMessageEvent;
emvcl_initdat.bConfigFilenameLen = strlen("V3CLVpTP_config.xml");
emvcl_initdat.pConfigFilename = "V3CLVpTP_config.xml";
ulRtn = EMVCL_Initialize(&emvcl_initdat);
```

2.2.2 EMVCL_ShowContactlessSymbol

Show contactless symbol on the screen.

PROTOTYPE	
ULONG	**EMVCL_ShowContactlessSymbol(void* pPara);**

PARAMETER(S)	
[IN] pPara	RFU. Please set to NULL.
RETURN VALUE	
d_EMVCL_NO_ERROR	
REMARK	
NONE	
EXAMPLE(S)	

2.2.3 EMVCL_ShowVirtualLED

Show Virtual LED on screen.

PROTOTYPE	
ULONG	**EMVCL_ShowVirtualLED(void* pPara);**
PARAMETER(S)	
[IN] pPara	RFU. Please set to NULL.
RETURN VALUE	
d_EMVCL_NO_ERROR	
REMARK	
NONE	
EXAMPLE(S)	

2.2.4 EMVCL_SetLED

Turn on/off the specific LED.

PROTOTYPE	
ULONG	**EMVCL_SetLED(BYTE bIndex, BYTE bOnOff);**
PARAMETER(S)	
[IN] bIndex	Bit mask. Bit0:LED1 Bit1:LED2 Bit2:LED3 Bit3:LED4 Bit4-Bit7:RFU Bit value 1 - Do the action specified by its

	corresponding bit in bOnOff. 0 – No action to do. Remain the current state.
[In] bOnOff	Bit mask - Bit0:action for LED1 Bit1:action for LED2 Bit2:action for LED3 Bit3:action for LED4 Bit4-Bit7:RFU Bit value 1 – Turn on 0 – Turn off
RETURN VALUE	
d_EMVCL_NO_ERROR	
d_EMVCL_RC_FAILURE	
d_EMVCL_RC_INVALID_PARAM	
REMARK	
NONE	
EXAMPLE(S)	

2.2.5 EMVCL_StartIdleLEDBehavior

LED behavior at idling depends on the UI Type. At the moment the contactless kernel is idling, use this function to start the corresponding LED behavior.

UI Type –

Normal UI: First indicator (blue led) is always on.

EUR UI: First indicator blink-on for approximately 200ms in every five seconds.

PROTOTYPE	
ULONG EMVCL_StartIdleLEDBehavior(void* pPara);	
PARAMETER(S)	
[IN] pPara	RFU. Please set to NULL.
RETURN VALUE	
d_EMVCL_NO_ERROR	
REMARK	
NONE	
EXAMPLE(S)	

2.2.6 EMVCL_StopIdleLEDBehavior

Stop idle LED behavior.

PROTOTYPE	
ULONG EMVCL_StopIdleLEDBehavior(void* pPara);	
PARAMETER(S)	
[IN] pPara	RFU. Please set to NULL.
RETURN VALUE	
d_EMVCL_NO_ERROR	
REMARK	
NONE	
EXAMPLE(S)	

2.2.7 EMVCL_SpecialEventRegister

Register special event (callback) function to EMVCL library. User can implement special event processing in their application and register it by this API. When the event is triggered by EMVCL kernel, the processing will go to the registered event and perform the implemented processing.

PROTOTYPE	
ULONG EMVCL_SpecialEventRegister(BYTE bEventID, void *pEventFunc);	
PARAMETER(S)	
[IN] bEventID	Register special event ID
[IN] pEventFunc	Register event (callback) function
RETURN VALUE	
d_EMVCL_NO_ERROR	
d_EMVCL_RC_INVALID_PARAM	
d_EMVCL_RC_INVALID_DATA	
REMARK	
NONE	
EXAMPLE(S)	

2.3 Setting Functions

This chapter introduces the setting APIs. The settings include TagCombination, CAPK, Parameter, Visa Capability, etc. Some setting has the maximum supported number as below:

Setting	Max Supported Num
TagCombination	64
CAPK	30

2.3.1 EMVCL_AIDSetTagData

Set tag data to the specific AID. EMVCL kernel can handle and store more tags settings for different combination list {TransType – AID - KERNELID} as the below table.

Transaction Type					
	AID1	AID2	AID3	….	AIDn
Kernel 1	TagsSetting1				
Kernel 2		TagsSetting2			
Kernel 3			TagsSetting3		
…..				……	
Kernel n					TagsSettingn

This API is used to set tag data for one combination list {TransType – AID-KERNEL ID}

PROTOTYPE

ULONG EMVCL_AIDSetTagData(BYTE bAction, EMVCL_AID_SET_TAG_DATA *stTagData);

PARAMETER(S)

[IN] Action	= 0x00 for one TagSetting Addition
	= 0x01 for one TagSetting Deletion
	= 0x02 for All TagSetting Deletion
[IN] stTagData	Point to a EMVCL_AID_TAG_DATA structure

	containing the tag setting to be set/deleted.
RETURN VALUE	
d_EMVCL_NO_ERROR	
d_EMVCL_RC_INVALID_PARAM	
d_EMVCL_RC_FAILURE	
REMARK	
NONE	
EXAMPLE(S)	

2.3.2 EMVCL_AIDGetTagData

Get tag data of the specific AID from EMVCL Contactless kernel.

PROTOTYPE	
ULONG EMVCL_AIDGetTagData(EMVCL_AID_GET_TAG_DATA *stGetTagData);	
PARAMETER(S)	
[OUT] stGetTagData	Point to a EMVCL_AID_GET_TAG_DATA structure to retrieve the corresponding Tag Setting.
RETURN VALUE	
d_EMVCL_NO_ERROR	
d_EMVCL_DATA_NOT_FOUND	
REMARK	
NONE	
EXAMPLE(S)	

2.3.3 EMVCL_SetCAPK

Set CA Public Key with specified RID to EMVCL Contactless kernel. The key to be set will belong to the specified RID.

PROTOTYPE	
ULONG EMVCL_SetCAPK(IN BYTE *baRID, IN EMVCL_CA_PUBLIC_KEY *stCAPubKey);	
PARAMETER(S)	
[IN] baRID	Point to a buffer containing RID. RID must be 5 bytes.
[IN] stCAPubKey	Point to a EMVCL_CA_PUBLIC_KEY structure

	containing the public key to be set.
RETURN VALUE	
d_EMVCL_NO_ERROR	
d_EMVCL_RC_INVALID_PARAM	
d_EMVCL_RC_FAILURE	
REMARK	
NONE	
EXAMPLE(S)	

2.3.4 EMVCL_GetCAPK

Get CA Public Key with specified RID from EMVCL Contactless kernel.

PROTOTYPE	
ULONG EMVCL_GetCAPK(IN BYTE *baRID, IN BYTE bKID, OUT EMVCL_CA_PUBLIC_KEY *stCAPubKey);	
PARAMETER(S)	
[IN] baRID	Point to a buffer containing RID. RID must be 5 bytes.
[IN] bKID	PK Index
[OUT] stCAPubKey	Point to a EMVCL_CA_PUBLIC_KEY structure to retrieve the corresponding CA public key.
RETURN VALUE	
d_EMVCL_NO_ERROR	
d_EMVCL_DATA_NOT_FOUND	
REMARK	
NONE	
EXAMPLE(S)	

2.3.5 EMVCL_ListAllCAPKID

List all CA Public Key Indexs with its RID

PROTOTYPE	
ULONG EMVCL_ListAllCAPKID(BYTE *baRBuf, UINT *uiRLen);	
PARAMETER(S)	
[OUT] baRBuf	Point to a buffer to retrieve CAPK Index list. The

	format is : Number of RID(s) – 1 byte RID – 5 bytes Number of Keys – 1 byte, noted as N1 Key Indexs – N1 byte(s) RID – 5 bytes Number of Keys – 1 byte, noted as N2 Key Indexs – N2 byte(s) ....
[OUT] uiRLen	Return the total length of CAPK ID list.
RETURN VALUE	
d_EMVCL_NO_ERROR	
REMARK	
NONE	
EXAMPLE(S)	

2.3.6 EMVCL_SetParameter

Set parameter data

PROTOTYPE		
ULONG		**EMVCL_SetParameter(EMVCL_PARAMETER_DATA *stPara);**
PARAMETER(S)		
[IN] stPara		Point to an EMVCL_PARAMETER_DATA structure containing the parameter data to be set.
Index	Len	
= 0x0002	= X(2)	Sale Timeout. default value is 0x3A98, 15000ms
= 0x100A	= X(1)	UI Type. 0x00-Normal(default), 0x01-EUR
= 0x100B	= X(1)	Visa EUR CL TIG Specification Follow. 0x00-NOT follow TIG rule(default), 0x01-Follow
RETURN VALUE		
d_EMVCL_NO_ERROR		
d_EMVCL_RC_INVALID_PARAM		
d_EMVCL_RC_FAILURE		
REMARK		
NONE		
EXAMPLE(S)		

2.3.7 EMVCL_GetParameter

Get one parameter data

PROTOTYPE		
ULONG EMVCL_GetParameter(UINT uiPID, EMVCL_PARAMETER_DATA *stPara);		
PARAMETER(S)		
[IN] uiPID		Parameter Index.
Index	Len	
= 0x0002	= X(2)	Sale Timeout.
= 0x100A	= X(1)	UI Type.
= 0x100B	= X(1)	Visa EUR CL TIG Specification Follow.
[OUT] stPara		Point to a EMVCL_PARAMETER_DATA structure to retrieve the specific parameter data value.
RETURN VALUE		
d_EMVCL_NO_ERROR		
d_EMVCL_RC_INVALID_PARAM		
REMARK		
NONE		
EXAMPLE(S)		

2.3.8 EMVCL_SetUIType

Set UI Type

PROTOTYPE	
ULONG EMVCL_SetUIType(BYTE bType);	
PARAMETER(S)	
[IN] bType	UI Type
= 0x00	Normal UI
= 0x01	EUR. UI
RETURN VALUE	
d_EMVCL_NO_ERROR	
d_EMVCL_RC_INVALID_PARAM	
d_EMVCL_RC_FAILURE	
REMARK	

NONE

EXAMPLE(S)

2.3.9 EMVCL_GetUIType

Get current UI type

PROTOTYPE

void EMVCL_GetUIType(BYTE *bType);

PARAMETER(S)

[OUT] bType	Current UI type
= 0x00	Normal UI
= 0x01	EUR UI

RETURN VALUE

NONE

REMARK

NONE

EXAMPLE(S)

2.3.10 EMVCL_VisaSetCapability

Visa has several contactless specification in the field, such as Wave1 card, MSD, Wave3 (qVSDC) card. This function is used to enable/disable to support such kind of Visa contactless cards by setting the corresponding scheme ID.

The supported scheme IDs are as below.

VISA_WAVE_QVSDC	0x17

PROTOTYPE

ULONG EMVCL_VisaSetCapability(EMVCL_SCHEME_DATA *stScheme, EMVCL_SCHEME_DATA *stRsp);

PARAMETER(S)

[IN] stScheme	Point to a EMVCL_SCHEME_DATA structure containing scheme ID(s). Action field with setting 0x01 indicates “Active” while 0x00 indicates “Deactive”.
[OUT] stRsp	Point to a EMVCL_SCHEME_DATA structure to return accepted scheme ID(s).

RETURN VALUE	
d_EMVCL_NO_ERROR	
d_EMVCL_RC_INVALID_SCHEME	
REMARK	
NONE	
EXAMPLE(S)	

2.3.11 EMVCL_VisaGetCapability

Get Visa Capability status.

PROTOTYPE	
ULONG EMVCL_VisaGetCapability(EMVCL_SCHEME_DATA *stScheme);	
PARAMETER(S)	
[OUT] stScheme	Point to a EMVCL_SCHEME_DATA structure to get the required schemeID(s) status
RETURN VALUE	
d_EMVCL_RC_SCHEME_SUPPORTED	
d_EMVCL_RC_INVALID_DATA	
d_EMVCL_RC_INVALID_SCHEME	
REMARK	
NONE	
EXAMPLE(S)	

2.4 Transaction Functions

2.4.1 EMVCL_StartTransactionEx

Start an EMV contactless transaction with transaction related data.
Get the transaction response when the transaction is finish.

Transaction related data should be TLV format.(TLV1 + TLV2 + .. + TLVn). ex : 9F0206000000015009F0306000000000000.... .

Transaction Related Data includes : (M:Madatory, O:Option, X:Not Supported)

Tag 9F02	(Amount Authorized(Numeric))	: visa - M	, MC - M
Tag 9F03	(Amount Other(Numeric))	: visa - O	, MC - O
Tag 9C	(Transaction Type)	: visa - M	, MC - O
Tag 9F53	(Transaction Category Code)	: visa - X	, MC - O
Tag 5F2A	(Transaction Currency Code)	: visa - M	, MC - O

Special Transaction Related Data for PayPass 3.0 Balance Reading function :
Tag DF8104 (Balance Read before Gen AC) :
MC with no Balance Reading - Absent , MC with Balance Reading - O
Tag DF8105 (Balance Read after Gen AC) :
MC with no Balance Reading - Absent , MC with Balance Reading - O

Forced Transaction Online :
Add Tag DF9F01 with value 01h (DF9F01 01 01) to Transaction Related Data

Application Prefer Order :
Add Tag DF9F02 to Transaction Related Data.
The data format is (1st PerferAIDLen + 1st PerferAID + 2nd PerferAIDLen + 2nd PerferAID + …+ nth PerferAIDLen + nth PerferAID).
PreferAIDLen : Prefer Application AID Len (1 Byte).
PreferAID : Prefer Application AID (5-16 Bytes).
Ex : if applications A0000000031010 and A0000000132020 are perfered to perform transaction, please add data
DF9F021007A000000003101007A0000000132020

PROTOTYPE

ULONG EMVCL_StartTransactionEx(EMVCL_ACT_DATA *stACTData, EMVCL_RC_DATA_EX *stRCDataEx);

PARAMETER(S)

[IN] stACTData	Point to a EMVCL_ACT_DATA structure to send transaction related data
[OUT] stRCDataEx	Point to a EMVCL_RC_DATA_EX structure to get the transaction data.

RETURN VALUE

d_EMVCL_RC_DATA
d_EMVCL_RC_FAILURE
d_EMVCL_RC_FALLBACK
d_EMVCL_TRY_AGAIN
d_EMVCL_TX_CANCEL
d_EMVCL_RC_MORE_CARDS
d_EMVCL_RC_NO_CARD

REMARK

NONE

EXAMPLE(S)

2.4.2 EMVCL_InitTransactionEx

Start an EMV contactless transaction with transaction related data only.
The transaction result must be retrieved by EMVCL_PerformTransactionEx function.

PROTOTYPE

ULONG EMVCL_InitTransactionEx(BYTE bTagNum, BYTE *baTransData, USHORT usTransDataLen);

PARAMETER(S)

[IN] bTagNum	Number of transaction related data.
[IN] baTransData	Transaction related data
[IN]usTransDataLen	Transaction related data length

RETURN VALUE

d_EMVCL_NO_ERROR
d_EMVCL_RC_FAILURE

REMARK

NONE

EXAMPLE(S)

2.4.3 EMVCL_PerformTransactionEx

This function is used to perform transaction and get the result of transaction issued by EMVCL_InitTransactionEx.

PROTOTYPE	
ULONG EMVCL_PerformTransactionEx(EMVCL_RC_DATA_EX *stRCDataEx);	
PARAMETER(S)	
[OUT] stRCDataEx	Point to struct EMVCL_RC_DATA_EX to get the transaction result.
RETURN VALUE	
d_EMVCL_RC_DATA	
d_EMVCL_RC_FAILURE	
d_EMVCL_RC_FALLBACK	
d_EMVCL_TRY_AGAIN	
d_EMVCL_TX_CANCEL	
d_EMVCL_RC_MORE_CARDS	
d_EMVCL_RC_NO_CARD	
d_EMVCL_PENDING	
d_EMVCL_RC_DEK_SIGNAL	
REMARK	
NONE	
EXAMPLE(S)	

2.4.4 EMVCL_PollTransactionEx

This is a backward compatible API. This API also can receive the transaction response. It works as EMVCL_PerformTransactionEx.

PROTOTYPE
ULONG EMVCL_PollTransactionEx(EMVCL_RC_DATA_EX *stRCDataEx,ULONG ulMS);
PARAMETER(S)

[OUT] stRCDataEx	Point to struct EMVCL_RC_DATA_EX to get the transaction result.
[IN] ulMS	RFU
RETURN VALUE	
The same with RETURN VALUE of EMVCL_PerformTransactionEx	
REMARK	
NONE	
EXAMPLE(S)	

2.4.5 EMVCL_CancelTransaction

Abort the current transaction. This function is used with EMVCL_InitTransactionEx and EMVCL_PerformTransactionEx. After calling this function, please call EMVCL_PerformTransactionEx or EMVCL_PollTransactionEx to check returning code. If the returning code of EMVCL_PerformTransactionEx or EMVCL_PollTransactionEx is d_EMVCL_TX_CANCEL, it indicates the transaction is canceled completely.

PROTOTYPE
ULONG EMVCL_CancelTransaction(void);
PARAMETER(S)
NONE
RETURN VALUE
d_EMVCL_NO_ERROR
REMARK
NONE
EXAMPLE(S)

2.4.6 EMVCL_AnalyzeTransactionEx

Analyze transaction Response Data EMV_RC_DATA_EX

PROTOTYPE	
void EMVCL_AnalyzeTransactionEx(EMVCL_RC_DATA_EX *stRCDataEx, EMVCL_RC_DATA_ANALYZE *stRCDataAnalyze);	
PARAMETER(S)	
[IN] stACTDataEx	Transaction response data

[OUT] stRCDataAnalyze	Point to struct EMVCL_RC_DATA_ANALYZ to show analyzed result.

RETURN VALUE

NONE

REMARK

NONE

EXAMPLE(S)

2.4.7 EMVCL_CompleteEx

Perform Complete/Issuer Update Processing.

PROTOTYPE

void EMVCL_CompleteEx(BYTE bAction, BYTE *baARC, UINT uiIADLen, BYTE *baIAD, UINT uiScriptLen, BYTE *baScript, EMVCL_RC_DATA_EX *stRCDataEx);

PARAMETER(S)

[IN] bAction	
= 0x01	Host responds to “APPROVED”
= 0x02	Host responds to “DECLINED”
= 0x03	Unable connect to Host
[IN] baARC	Authorization Response Code, Tag 8A, This value must be 2 ASCII chars.
[IN] uiIADLen	Length of Issuer Authentication Data (8 to 16 bytes)
[IN] baIAD	Issuer Authentication Data, Tag 91. Please only input the data of Tag 91. Only data (no TL).
[IN] uiScriptLen	Length of issuer script data
[IN] baScript	Issuer script data. Pointer to the emitter scripts, composed by one or more ‘71’ or ‘72’ concatenated templates (complete TLV structure).
[OUT] stRCDataEx	Point to struct EMVCL_RC_DATA_EX to get the transaction result.

RETURN VALUE

d_EMVCL_RC_DATA
d_EMVCL_RC_FAILURE
d_EMVCL_RC_FALLBACK
d_EMVCL_TRY_AGAIN
d_EMVCL_TX_CANCEL

d_EMVCL_RC_MORE_CARDS
d_EMVCL_RC_NO_CARD
d_EMVCL_RC_DEK_SIGNAL

REMARK

NONE

EXAMPLE(S)

2.4.8 EMVCL_DetectTransactionEx

This API is used to detect card application which is selected to perform transaction. When calling this API, emvcl kernel will request to present card. After card detected, application informations (Select PPSE Response, Selected AID, Select AID Response) will be returned. In this phase, transaction has not been finished yet. To continue transaction, please call EMVCL_InitTransactionEx + EMVCL_PerformTransactionEx. To stop transaction, please call EMVCL_CancelTransaction.

PROTOTYPE

ULONG EMVCL_DetectTransactionEx(EMVCL_DETECT_TXN_DATA *pstDetectTxnData);

PARAMETER(S)

[INOUT] pstDetectTxnData	Point to struct EMVCL_DETECT_TXN_DATA

RETURN VALUE

d_EMVCL_RC_DATA
d_EMVCL_RC_FAILURE
d_EMVCL_TX_CANCEL
d_EMVCL_RC_MORE_CARDS
d_EMVCL_RC_NO_CARD

REMARK

NONE

EXAMPLE(S)

```
ulRtn = EMVCL_DetectTransactionEx();
if(isDetectTxnDataOK == TRUE)
{
    //Continue transction
    EMVCL_InitTransactionEx();
    EMVCL_PerformTransactionEx();
    ……
    //transaction finish
```

```
        }
        else
        {
            //Stop transaction and then return
            EMVCL_CancelTransaction();
            return ;
        }
```

Transaction Functions Notes

Transaction Note 1 :

This note is information for EMVCL_InitTransactionEx() and EMVCL_PerformTransactionEx().

EMVCL_InitTransactionEx() and EMVCL_PerformTransactionEx() are couple of APIs. They should be used together.

EMVCL_InitTransactionEx() will use the Transaction Related Data to perform Pre-Processing. If its returned code is d_EMVCL_NO_ERROR, it means there are suited applications in the emvcl kernel and please call EMVCL_PerformTransactionEx() to perform transaction.

EMVCL_PerformTransactionEx() will perform a transaction and return the outcome from emvcl kernel.

If EMVCL_PerformTransactionEx() requested a Restart/Try Again outcome, please call EMVCL_PerformTransactionEx() again to restart another transaction. The Transaction Related Data (ex. Amount, Transaction Type..) will be kept in emvcl kernel. So the 2nd transaction will be started with the same condition of 1st Transaction.

Transaction Note 2 :

This note is for how to implement Visa qVSDC Issuer Update Processing.

Issuer Update Processing will be performed after Online authorization. During Issuer Update Processing, the EXTERNAL AUTHENTICATE and SCRIPT command may be performed depending on the authentication data form host. If terminal supports Issuer update processing and host response IAD(Issuer Authentication Data) or Script data, the Issuer Update Processing should be performed and cardholder is instructed to present their card once more.

The implement flow will be below :

1. Before transaction, setting TTQ byte 3 bite 8 to 1b terminal supports Issuer Update Processing.

2. Starting a transaction by calling EMVCL_StartTransactionEx() or EMVCL_InitTransactionEx()+EMVCL_PerformTransactionEx().

3. Transaction outcome should be online.

4. Sending online data to host and wait for response.

5. If Host responds Issuer Authentication Data or Scripts, please go to step 6. If no data returned, please go to step 8.

6. Calling EMVCL_CompleteEx() to perform Issuer update processing.

7. Present Card again.

8. Transaction is completed.

Transaction Note 3 :

This note is information for EMVCL_CompleteEx().

The rule for getiing the response data of EMVCL_CompleteEx() is the same with EMVCL_PerformTransactionEx(). Developer can set Upload DOL Tags to get the data objects. The Upload DOL Tags are used for both EMVCL_CompleteEx() and EMVCL_PerformTransactionEx(). If Upload DOL Tags are not set, emvcl kernel will return the default values.

When calling EMVCL_CompleteEx(), emvcl kernel will according the input data to perform the corresponding procedure. If there are IAD or Scripts from host and input to this API, emvcl kernel may perform EXTERNAL AUTHENTICATE or SCRIPT or GenAC2 command. If there is only the the ARC (Authorisation Response Code) from host (ie. parameter bAction and ARC are present, IAD len = 0, script len = 0), emvcl kernel will according the bAction to update the outcome and then refer to Upload DOL Tag to return the transaction response data.

Transaction Note 4 :

This note is information for Issuer Script Results.

There is one Tag to indicate the Issuer Script Results. After performing the scripts command, emvcl kernel will generate the Issuer Script Results Tag. Developer can request the Issuer Script Results Tag in Upload DOL Tag and get it in the response of EMVCL_CompleteEx().

3 *Special Event*

3.1 Special Event Table

Code	Value	Description
d_EMVCL_EVENTID_LED_PIC_SHOW	0x01	Control LED1-4 behaviors

3.2 Special Event Function

3.2.1 EVENT_EMVCL_LED_PIC_SHOW

Control LED 1-4 behaviors

PROTOTYPE

typedef ULONG (*EVENT_EMVCL_LED_PIC_SHOW)(BYTE bIndex, BYTE bOnOff);

PARAMETER(S)

[IN] bIndex	Bit mask. Bit0:LED1 Bit1:LED2 Bit2:LED3 Bit3:LED4 Bit4-Bit7:RFU Bit value 1 - Do the action specified by its corresponding bit in bOnOff. 0 – No action to do. Remain the current state.
[In] bOnOff	Bit mask - Bit0:action for LED1 Bit1:action for LED2 Bit2:action for LED3 Bit3:action for LED4

Bit4-Bit7:RFU

Bit value

1 – Turn on

0 – Turn off

RETURN VALUE

d_EMVCL_NO_ERROR

d_EMVCL_RC_FAILURE

REMARK

NONE

EXAMPLE(S)

4 Returning Code Table

Code	Value	Description
d_EMVCL_NO_ERROR	0x00000000	No error
d_EMVCL_PENDING	0x80000020	Pending
d_EMVCL_TX_CANCEL	0x80000021	Transaction is canceled
d_EMVCL_INIT_TAGSETTING_ERROR	0x80000022	Init TagSetting Error
d_EMVCL_INIT_CAPK_ERROR	0x80000023	Init CAPK data Error
d_EMVCL_INIT_PARAMETER_ERROR	0x80000024	Init Parameter Error
d_EMVCL_INIT_CAPABILITY	0x80000025	Init Capabaility data Error
d_EMVCL_INIT_REVOCATION_ERROR	0x80000026	Init Revocation data Error
d_EMVCL_INIT_EXCEPTION_FILE_ERROR	0x80000027	Init Exception File Error
d_EMVCL_DATA_NOT_FOUND	0x80000030	Data is NOT found
d_EMVCL_TRY_AGAIN	0x800000051	Try Again
d_EMVCL_RC_DATA	0xA0000001	Indicate the Transaction Data Received
d_EMVCL_RC_SCHEME_SUPPORTED	0xA0000004	The payment scheme is supported
d_EMVCL_RC_FAILURE	0xA00000FF	General failure.
d_EMVCL_RC_INVALID_DATA	0xA00000F8	Invalid Data
d_EMVCL_RC_INVALID_PARAM	0xA00000F7	No such parameters.
d_EMVCL_RC_INVALID_SCHEME	0xA00000F5	Invalid Scheme
d_EMVCL_RC_MORE_CARDS	0xA00000F3	More than 1 card
d_EMVCL_RC_NO_CARD	0xA00000F2	Contactless card is not present.
d_EMVCL_RC_FALLBACK	0xA00000E9	Fall back
d_EMVCL_RC_DEK_SIGNAL	0xA00000E0	DEK Signal

5 User Req Data Informatiom

bMessageIdentifier	**Value**
d_EMVCL_USR_REQ_MSG_CARD_READ_OK	0x17
d_EMVCL_USR_REQ_MSG_TRY_AGAIN	0x21
d_EMVCL_USR_REQ_MSG_APPROVED	0x03
d_EMVCL_USR_REQ_MSG_APPROVED_SIGN	0x1A
d_EMVCL_USR_REQ_MSG_DECLINED	0x07
d_EMVCL_USR_REQ_MSG_ERROR_OTHER_CARD	0x1C
d_EMVCL_USR_REQ_MSG_INSERT_CARD	0x1D
d_EMVCL_USR_REQ_MSG_SEE_PHONE	0x20
d_EMVCL_USR_REQ_MSG_AUTHORISING_PLEASE_WAIT	0x1B
d_EMVCL_USR_REQ_MSG_CLEAR_DISPLAY	0x1E
d_EMVCL_USR_REQ_MSG_ENTER_PIN	0x09
d_EMVCL_USR_REQ_MSG_PROCESSING_ERR	0x0F
d_EMVCL_USR_REQ_MSG_REMOVE_CARD	0x10
d_EMVCL_USR_REQ_MSG_WELCOME	0x14
d_EMVCL_USR_REQ_MSG_PRESENT_CARD	0x15
d_EMVCL_USR_REQ_MSG_PROCESSING	0x16
d_EMVCL_USR_REQ_MSG_INSERT_OR_SWIPE_CARD	0x18
d_EMVCL_USR_REQ_MSG_PRESENT_1_CARD_ONLY	0x19
d_EMVCL_USR_REQ_MSG_PRESENT_CARD_AGAIN	0x21
d_EMVCL_USR_REQ_MSG_NO_CARD	0xA0
d_EMVCL_USR_REQ_MSG_NA	0xFF

bMessageIdentifier	**Value**
d_EMVCL_USR_REQ_STATUS_NOT_READY	0x00
d_EMVCL_USR_REQ_STATUS_IDLE	0x01
d_EMVCL_USR_REQ_STATUS_READY_TO_READ	0x02
d_EMVCL_USR_REQ_STATUS_PROCESSING	0x03
d_EMVCL_USR_REQ_STATUS_CARD_READ_SUCCESSFULLY	0x04
d_EMVCL_USR_REQ_STATUS_PROCESSING_ERROR	0x05
d_EMVCL_USR_REQ_STATUS_NA	0xFF