

EMV Level 2 Library
API Manual
V 1.55

Evolution of the API:

API Version	Date	Descriptions
1.0	2004.05.31	Created
1.1	2004.11.22	Add PINRES_FAILBLOCKED to APP_PINVerifyResult
1.2	2006.11.20	Add VEGA7000 Modify compiler requirement
1.3	2010.07.02	Add APP_iIDCardVerify()
1.4	2012.08.07	Add LastErrorCode definition. Modify Product Descriptions.
1.5	2012.09.10	Add "Last Error Code" to PP functions description table.
1.51	2012.10.01	Add additional returning code to section 6.1
1.52	2012.10.02	Add additional Last Error code to section 4
1.53	2012.10.11	I. Modify input value (advice) of APP_ShowAdvice. II. Remove Advice Message table. III. Add additional FLG_MF_ACCEPT description to section 4.10 (PP_iTermActionAnalysis).
1.54	2012.10.19	1. Add notation to Chapter 4 and section 4.15. 2. Add additional returning codes for PP_iXXX functions.
1.55	2012.11.08	1. Add description for d_PP_LASTERR_ICC_CERT_FORMAT_ERROR in Last Error Code table. 2. Remove d_PP_LASTERR_CAS_EXTERNAL_AUTH_FAIL from section 4.11(PP_iCardActionAnalysis).

WARNING	0
ABOUT THIS MANUAL	0
1. ABOUT THIS MANUAL	1
2. SYSTEM REQUIREMENT	1
3. EMV LV2 LIBRARY SPECIFICATION.....	1
4. API TO BE SUPPLIED BY THE KERNEL.....	2
4.1 PP_EMV_GetVERSION	2
4.2 PP_EMV_RESET	2
4.3 PP_ISCDEFAPP.....	2
4.4 PP_IGetCANDIDATELIST.....	4
4.5 PP_ISELECT_APP.....	6
4.6 PP_IAUTHENTICATION	8
4.7 PP_IPROCESSINGRESTRICTIONS.....	10
4.8 PP_ICARDHOLDERVERIFICATION	11
4.9 PP_IRiskMANAGEMENT	13
4.10 PP_ITERMACTIONANALYSIS.....	14
4.11 PP_ICARDACTIONANALYSIS.....	15
4.12 PP_IDEFTERMACTIONANALYSIS	17
4.13 PP_ICOMPLETION	17
4.14 PP_IEMVGETDATA	21
4.15 PP_IGetLASTERROR	21
5. FUNCTIONS TO BE SUPPLIED BY THE APPLICATION	22
5.1 APP_FGetCAPK	22
5.2 APP_FCHECKREVOCATION.....	23
5.3 APP_FGetTERMDATA	23
5.4 APP_IGetPINOffline.....	23
5.5 APP_IGetPINOnline.....	24
5.6 APP_PINVERIFYRESULT	24
5.7 APP_SHOWADVICE.....	25
5.8 APP_IDCARDVERIFY	26
6. APPENDIX. DATA DEFINITION	29
6.1 RETURN CODE.....	29
6.2 AUC FLAGS	29
6.3 TAA FLAGS	29
6.4 ONLINE FLAGS	29

6.5	PIN VERIFY DISPLAY FLAGS	29
6.6	LAST ERROR CODES	30

WARNING

Information in this document is subject to change without prior notice.

No part of this publication may be reproduced, transmitted, stored in a retrieval system, nor translated into any human or computer language, in any form or by any means, electronic, mechanical, magnetic, optical, chemical, manual, or otherwise, without the prior written permission of Castles Technology Co., LTD.

All trademarks mentioned are proprietary of their respective owners.

ABOUT THIS MANUAL

This manual describes the application programming interface (API) functions; the application software developers should refer to this manual to develop their own software to drive the VEGA5000(S), VEGA7000, VEGA9300, and EMV8800 terminal.

1. About This Manual

The VEGA5000(S), VEGA7000, VEGA9300, and EMV8800 included certificated EMV Level 2 kernel. It provides the complete function for the transaction of EMV LV2, and thus greatly simplifies the procedure of the transaction. This document details all the EMV LV2 APIs and implement information you need.

2. System Requirement

IBM PC Compatible

CASTLES DEVELOPER SDK for Vega5000(S)

CASTLES DEVELOPER SDK for Vega5000(S)

ARM Developer Suite v1.2 for Vega9000

Keil Compiler uVersion 2 PK51 for EMV8800

1 available RS232 port

Windows 95/98/ME/NT/2000/XP

VEGA5000(S) / VEGA9300 / VEGA7000 / EMV8800 Terminal

3. EMV LV2 Library Specification

The EMV LV2 kernel is EMV 4.2 approval and support the following EMV 4.2 standards:

Select application

Cardholder Confirmation

PSE

Explicit Selection

Offline data authentication:

Certification Authority Public Key with a Modulus of 248 bytes

SDA

DDA

DDA/AC

CVM

Support PIN bypassing.

Plaintext Offline PIN

Enciphered Offline PIN

Enciphered Online PIN

Signature

Online

Support Advice.

Support Approval of the transaction is forced.

Support card initial referral and issuer initial referral.

Others

Exception list.
Transaction log.
Revocation list.

4. API to Be Supplied by the Kernel

This item describes the functions made available by the kernel for the processing of cards EMV.

Note that, only if the returning code of PP_xxx function is PP_OK, the processing of transaction can be continued. Namely, if the returning code of PP_iXXX function is PP_OK, the transaction should be continued, while if the returning code of PP_iXXX function is not PP_OK, the transaction should be aborted.

4.1 PP_EMV_GetVersion

Get EMV LV2 kernel version number.

<i>Syntax C</i>	void PP_EMV_GetVersion (char *ver)	
<i>Input Data</i>	None	
<i>Output Data</i>	ver	8 bytes version number
<i>Return Data</i>	None	
<i>Return Code</i>	PP_OK	No error found.
<i>Sample Code</i>	BYTE baVersion[8]; PP_EMV_GetVersion (baVersion);	

:

4.2 PP_EMV_Reset

This function initiate the EMV kernel environment.

<i>Syntax C</i>	void PP_EMV_Reset (void)	
<i>Input Data</i>	None	
<i>Output Data</i>	None	
<i>Return Code</i>	None	
<i>Sample Code</i>	PP_EMV_Reset ();	

4.3 PP_iSCDefApp

This function inserts an application known by the terminal in the list of the

applications that can participate in the selection process in the card.

PP_iSCDefApp can be called several times, being this initiate process for PP_iSCDefApp (NULL).

Syntax C	short PP_iSCDefApp (DefAppStruct *pstData) typedef struct { ULONG ulRef; BYTE *pbAID; BYTE bAIDLen; BYTE bApplication_Selection_Indicator; } DefAppStruct;	
Input Data	pstData	Pointer for a structure with the data of the application to insert in the list, or NULL to begin the process. Code reference proprietor to be returned by the function PP_iGetCandidateList for each application recognized as “candidate” to the selection. Pointer for the Identifier of the application, from 5 to 16 bytes.
	ulRef	AID length
	pbAID	
	bAIDLen	
	bApplication_Selection_Indicator	
	0	Multiple occurrence supported
	1	One occurrence supported
Output Data	None	
Return Code	PP_OK	No error found
	PP_DATA_BUFFER_EXCEEDED	Buffer overflow of the storage.
	PP_INVALID_PARA	AID length incorrect.
Get Last Error Code	Null	

<i>Sample Code</i>	<pre> DefAppStruct stAppData; BYTE baAID[7]; short rtn; PP_iSCDefApp(NULL); //Clear memcpy(baAID, "\xA0\x00\x00\x00\x03\x10\x10", 7); stAppData.pbAID = baAID; stAppData.ulRef = 1000; stAppData.pbAID = baAID; stAppData.bAIDLen = 7; stAppData.bApplication_Selection_Indicator = 0; rtn = PP_iSCDefApp(&stAppData); if (rtn != PP_OK) return FALSE; </pre>
--------------------	--

4.4 PP_iGetCandidateList

This function should be called by the application after the insert of the list of applications being used calls to the function **PP_iSCDefApp**.

In **PP_iGetCandidateList**, the kernel talks to the card and it makes the process of obtaining of the candidates' list, be through PSE or direct selection. The labels of the applications candidates are returned, together with their proprietary reference codes, already in priority order so that the application can exhibit a menu.

In case only one candidate application exists, **pfConfirm** indicates if it should be confirmed by the cardholder of the card.

IMPORTANT: The kernel should obtain the Additional Terminal Capabilities (using **APP_fGetTermData**) in a way to use Application Preferred Names instead of Application Labels when Issuer Code Table Index is compatible.

After the call of this function, the main application should call **PP_iSCDefApp** to inform to the kernel the selected option.

<i>Syntax C</i>	<pre> short PP_iGetCandidateList(WORD *puiNumItems, char *vszLabels[], ULONG vulRef[], char *pfConFirm, char bNewSel) </pre>
<i>Input Data</i>	<p>puiNumItems Maximum candidate buffer supported.</p> <p>bNewSel When bNewSel is TRUE, reload the candidate list from card, otherwise only remove the failed selection application and refresh the candidate content. list</p>

<i>Output Data</i>	<p>puiNumItems Number of candidatas.</p> <p>vszLabels Strings (NULL ended) Vector for the application labels to be showed in the menu.</p> <p>vulRef Vector to the proprietary reference codes corresponding to the labels to be presented in the menu, for use of the main application.</p> <p>pfConFirm When puiNumItems is 1 (one), Must return TRUE if card bearer confirmation is necessary.</p>	
<i>Return Code</i>	<p>PP_OK No error found.</p> <p>PP_CRITICAL_MISTAKES Critical error need to terminate the transaction.</p> <p>PP_DATA_BUFFER_EXCEEDED Card application number is large than candidate list buffer in kernel.</p> <p>PP_TERMINAL_DATA_MISSING</p> <p>PP_FUNCTION_NOT_SUPPORTED</p> <p>PP_ONLY_1_AP_NO_FALLBACK</p>	
<i>Get Last Error Code</i>	<p>d_PP_LASTERR_FUNCTION_NOT_SUPPORTED 0x0001</p> <p>d_PP_LASTERR_CANDIDATE_LIST_FULL 0x0002</p> <p>d_PP_LASTERR_SEND_APDU_CMD_FAIL 0x0004</p> <p>d_PP_LASTERR_ONLY_1_AP_BLOCKED 0x0026</p>	

<p><i>Sample Code</i></p>	<pre> WORD NumItems; char *vszLabels[50]; char vszLabelstr[50][17]; ULONG vulRef[50]; char ConFirm; *** Put applications use PP_iSCDefApp *** for (i = 0; i < 50; i ++) { vszLabels[i] = vszLabelstr[i]; memset(vszLabels[i], 0, 17); } NumItems = 50; rtn = PP_iGetCandidateList(&NumItems, vszLabels, vulRef, &ConFirm, 1); if (rtn == PP_OK) { *** Show Application List *** if (NumItems == 0) return d_G9_STATUS_NO_AP_FOUND; } else return rtn; </pre>
---------------------------	--

4.5 PP_iSelect_App

After the presentation of the candidates' menu, the main application should call **PP_iSelectApp** to inform the selected application.

In PP_iSelectApp the kernel EMV should execute the following processing's governed by the EMV standard:

Get Processing Options; and
Read Application Data

In case the kernel identifies a problem that demands the retreat of the application from the candidates' list for a new selection, this situation must be informed by a return code so that the main application calls **PP_iGetCandidateList** again.

<i>Syntax C</i>	short PP_iSelect_App (short iAppldx)	
<i>Input Data</i>	iAppldx	Index of the selected application (from 0 and on), as the sequence returned by PP_iGetCandidateList .
<i>Output Data</i>	None	
<i>Return Code</i>	PP_OK PP_SELECTION_FAIL PP_CRITICAL_MISTAKES PP_IAP_9481_FALLBACK	No error found. Selection Failed, application must be removed from the menu (PP_iGetCandidateList should be called again to obtain a new menu). Critical error need to terminate the transaction.
<i>Get Last Error Code</i>	d_PP_LASTERR_APPLICATION_NOT_ALLOW 0x0007 d_PP_LASTERR_IAP_AIP_AFL_ERROR 0x0008 d_PP_LASTERR_IAP_UNKNOW_SW12 0x0009 d_PP_LASTERR_IAP_MISS_PAN 0x000A d_PP_LASTERR_IAP_MISS_CDOL1 0x000B d_PP_LASTERR_IAP_MISS_CDOL2 0x000C d_PP_LASTERR_IAP_APPLICATION_NOT_ALLOW 0x000D d_PP_LASTERR_MISS_AIP_AFL 0x0037 d_PP_LASTERR_DGP_DIS_TLV_FAIL 0x0039 d_PP_LASTERR_MULTIPLE_OCCURENCE 0x003A d_PP_LASTERR_CVM_DATA_LEN_LESS_8 0x003B d_PP_LASTERR_SFI_TEMPLATE_RULE_WRONG 0x007C d_PP_LASTERR_IAP_BAD_PADDING 0x0081 d_PP_LASTERR_IAP_FALLBACK_9481 0x0088	

<i>Sample Code</i>	<pre> rtn = PP_iGetCandidateList(&NumItems, vszLabels, vulRef, &ConFirm, 1); if (rtn == PP_OK && NumItems > 0) { rtn = PP_iSelect_App(0); //Select first application in candidate list if (rtn == PP_SELECTION_FAIL) { //Re-select the next application rtn = PP_iGetCandidateList(&NumItems, vszLabels, vulRef, &ConFirm, 0); } } </pre>
--------------------	--

4.6 PP_iAuthentication

This function executes the Offline Data Authentication procedure governed by the EMV standard. If necessary, the kernel should obtain the public key used in the authentication through the function **APP_fGetCAPK**.

During the authentication process, the validity of Issuer Public Key Certificate can be verified by the kernel through the function **APP_fCheckRevocation**.

<i>Syntax C</i>	short PP_iAuthentication (BYTE *pbDefDDOL, WORD uiDefDDOLLen);	
<i>Input Data</i>	pbDefDDOL	Pointer to "default" terminal DDOL.
	uiDefDDOLLen	Size in bytes of the "default" DDOL. In case this information doesn't exist, this value should be 0 (zero).
<i>Output Data</i>	None	
<i>Return Code</i>	PP_OK	No error found.
	PP_CRITICAL_MISTAKES	Critical error need to terminate the transaction.

Get Last Error Code	d_PP_LASTERR_ERROR_9F4A_RULE	0x000E
	d_PP_LASTERR_KEY_NO_FOUND	0x0010
	d_PP_LASTERR_PAN_NOT_SAME	0x0011
	d_PP_LASTERR_DDOL_MISS	0x0012
	d_PP_LASTERR_INTERNAL_AUTHENTICATE_FAIL	0x0013
	d_PP_LASTERR_SDA_LENGTH_NOT_MATCH	0x0014
	d_PP_LASTERR_NO_OFFLINE_DATA_AUTH_MATCH	0x0015
	d_PP_LASTERR_READ_DATA_TAG_NOT_70	0x0016
	d_PP_LASTERR_DIS_TLV_TAG_ZERO	0x0032
	d_PP_LASTERR_DIS_TLV_EXCEED_MAX_LEN	0x0035
	d_PP_LASTERR_DDOL_NOT_HAVE_9F37	0x0038
	d_PP_LASTERR_SDA_DATA_ERROR	0x003C
	d_PP_LASTERR_SDA_ALGORITHM_NOT_SUPPOR	0x003D
	d_PP_LASTERR_DDA_DATA_ERROR	0x003E
	d_PP_LASTERR_KEY_LENGTH_ERROR	0x0043
	d_PP_LASTERR_ISSUER_CERT_NOT_EXIST	0x0047
	d_PP_LASTERR_ISSUER_CERT_FORMAT_ERROR	0x0048
	d_PP_LASTERR_ISSUER_CERT_IIN_PAN_NOT_SAME	0x0049
	d_PP_LASTERR_ISSUER_CERT_REVOCATION_FOUND	0x004A
	d_PP_LASTERR_ISSUER_CERT_ALGORITHM_NOT_SUP	0x004B
	PORT	
	d_PP_LASTERR_ISSUER_CERT_LENGTH_ERROR	0x004C
	d_PP_LASTERR_ISSUER_CERT_EXPIRATION_DATE	0x004D
	d_PP_LASTERR_ISSUER_CERT_HASH_NOT_MATCH	0x004E
	d_PP_LASTERR_ISSUER_CERT_EXPONENT_NOT_EXIST	0x004F
	d_PP_LASTERR_ISSUER_CERT_REMAINDER_MISSING	0x0050
	d_PP_LASTERR_ICC_CERT_NOT_EXIST	0x0051
	d_PP_LASTERR_ICC_CERT_FORMAT_ERROR	0x0052
	d_PP_LASTERR_ICC_CERT_ALGORITHM_NOT_SUPPOR	0x0053
	T	
	d_PP_LASTERR_ICC_CERT_LENGTH_ERROR	0x0054
	d_PP_LASTERR_ICC_CERT_HASH_NOT_MATCH	0x0055
	d_PP_LASTERR_ICC_CERT_EXPIRATION_DATE	0x0056
	d_PP_LASTERR_ICC_CERT_EXPONENT_NOT_EXIST	0x0057
	d_PP_LASTERR_ICC_ISSUER_PK_NOT_EXIST	0x0058
	d_PP_LASTERR_ICC_CERT_REMAINDER_MISSING	0x0059
	d_PP_LASTERR_ISSUER_CERT_CAPKI_NOT_EXIST	0x0079
	d_PP_LASTERR_MISSING_TERMINAL_DATA	0x0075

<i>Sample Code</i>	<pre> BYTE DefaultDDOL[20]; memcpy(DefaultDDOL, "\x9F\x37\x04", 3); rtn = PP_iAuthentication(DefaultDDOL, 3); if (r != PP_OK) { return d_STATUS_CRITICAL_MISTAKES; } </pre>
--------------------	--

4.7 PP_iProcessingRestrictions

This function executes the following steps governed by the EMV standard:

Application Version Number (it Verifies the version of the application, comparing it with one of the entrances of the supplied vector)

Application Usage Control (it Verify the restrictions of use of the application based in the information supplied in **bFlags**).

Application Effective/Expiration Dates Checking (it Verifies application dates, comparing them with Transaction Dates '9A).

<i>Syntax C</i>	short PP_iProcessingRestrictions(short iAppVerNum, BYTE vvbAppVerList[][2], BYTE bFlags)	
<i>Input Data</i>	iAppVerNum vvbAppVerList bFlags	Number of accepted versions. Accepted Versions list, containing iAppVerNum items. Flags to be used by <i>Application Usage Control</i> , It is a combination of the constants: USCTRL_ATM USCTRL_CASHBACK USCTRL_GOODS USCTRL_SERVICES
<i>Output Data</i>	None	
<i>Return Code</i>	PP_OK PP_CRITICAL_MISTAKES	No error found. Critical error need to terminate the transaction.
<i>Get Last Error Code</i>	d_PP_LASTERR_MISSING_TERMINAL_DATA d_PP_LASTERR_MISS_APP_EXPIRATION_DATE	0x0075 0x0017

<i>Sample Code</i>	<pre> BYTE vvbAppVerList[1][2]; memcpy(&vvbAppVerList[0][0], "\x02\x00", 2); rtn = PP_iProcessingRestrictions(1, vvbAppVerList, USCTRL_SERVICES USCTRL_GOODS); if (tnr != PP_OK) { return d_STATUS_CRITICAL_MISTAKES; } </pre>
--------------------	--

4.8 PP_iCardholderVerification

This function executes the Cardholder Verification step, governed by the EMV standard.

In case the kernel needs a capture of PIN, the functions **APP_iGetPINOnline** and **APP_iGetPINOffline** can be called by him.

In the case of PIN offline, the function **APP_PINVerifResult** should be called by the kernel to inform the result of the verification to the main application (this can be made several times while typed PIN is considered invalid by the card).

<i>Syntax C</i>	short PP_iCardholderVerification(char *pfSignature)	
<i>Input Data</i>	None	
<i>Output Data</i>	pfSignature	Signature in paper should be obtained.
<i>Return Code</i>	PP_OK PP_CRITICAL_MISTAKES	No error found. Critical error need to terminate the transaction.
<i>Get Last Error Code</i>	d_PP_LASTERR_MISSING_TERMINAL_DATA d_PP_LASTERR_CARDHOLDER_VER_NOT_SUPP d_PP_LASTERR_CVM_LIST_MISSING d_PP_LASTERR_CVM_FINISH d_PP_LASTERR_CVM_PLAIN_TEXT d_PP_LASTERR_CVM_ENC_PIN_ONLINE d_PP_LASTERR_CVM_PLAINTEXT_PIN_SIGNATURE d_PP_LASTERR_CVM_ENC_PIN d_PP_LASTERR_CVM_ENC_PIN_SIGNATURE_OFFLINE d_PP_LASTERR_CVM_SIGNATURE d_PP_LASTERR_CVM_IDCARD_VERIFY	0x0075 0x0018 0x0019 0x001A 0x001B 0x001C 0x001D 0x001E 0x001F 0x0020 0x0025

	d_PP_LASTERR_PIN_CERT_NOT_EXIST 0x005A d_PP_LASTERR_PIN_CERT_LENGTH_ERROR 0x005B d_PP_LASTERR_PIN_CERT_FORMAT_ERROR 0x005C d_PP_LASTERR_PIN_CERT_ALGORITHM_NOT_SUPPORT 0x005D d_PP_LASTERR_PIN_CERT_HASH_NOT_MATCH 0x005E d_PP_LASTERR_PIN_CERT_EXPIRATION_DATE 0x005F d_PP_LASTERR_PIN_CERT_KEY_LENGTH_ERROR 0x0060 d_PP_LASTERR_PIN_CERT_EXP_NOT_EXIST 0x0085 d_PP_LASTERR_READ_PIN_TRY_COUNT_FAIL 0x0061 d_PP_LASTERR_CVM_PLAIN_TEXT_PIN_NOT_KEYIN 0x0062 d_PP_LASTERR_CVM_PLAIN_TEXT_PIN_TRY_LIMIT_EXCEED 0x0063 ED d_PP_LASTERR_CVM_PLAIN_TEXT_PIN_OK 0x0064 d_PP_LASTERR_CVM_PLAIN_TEXT_PIN_WRONG 0x0066 d_PP_LASTERR_CVM_PLAIN_TEXT_UNKNOW_SW12 0x0067 d_PP_LASTERR_CVM_TERMINAL_NOT_SUPPORT_SPECIFY_CVM 0x0068 CVM d_PP_LASTERR_CVM_ENC_PIN_ONLINE_PIN_NOT_KEYIN 0x0069 d_PP_LASTERR_CVM_ENCIPHERED_PIN_NOT_KEYIN 0x006C d_PP_LASTERR_CVM_ENCIPHERED_PIN_TRY_LIMIT_EXCEED 0x006D DED d_PP_LASTERR_CVM_ENCIPHERED_PIN_UNKNOW_SW12 0x006E d_PP_LASTERR_CVM_ENCIPHERED_PIN_OK 0x006F d_PP_LASTERR_CVM_ENCIPHERED_PIN_WRONG 0x0070 d_PP_LASTERR_CVM_ENCIPHERED_PIN_GET_RN_UNKNOW_SW12 0x0071 _SW12 d_PP_LASTERR_CVM_PLAIN_TEXT_PIN_BLOCK 0x007D d_PP_LASTERR_CVM_ENCIPHERED_PIN_BLOCK 0x007E
Sample Code	<pre> BYTE isSignature; rtn = PP_iCardholderVerification(&isSignature); if (r != PP_OK) { return d_STATUS_CRITICAL_MISTAKES; } if (isSignature == 1) //Print receipt </pre>

4.9 PP_iRiskManagement

This function executes the following steps governed by the EMV standard:

Floor limit checking

Random transaction selection

<i>Syntax C</i>	short PP_iRiskManagement(ULONG ulLastAmount, BYTE bTargetPerc, BYTE bMaxTargetPerc, ULONG ulThresholdValue)	
<i>Input Data</i>	ulLastAmount	Value of the last transaction stored in the terminal log for the same card (to be used in Floor Limit Checking).
	bTargetPerc	Target Percentage to be Used for Random Selection (from 0 to 99).
	bMaxTargetPerc	Maximum Target Percentage to be Used for Biased Random Selection (from 0 to 99).
	ulThresholdValue	Threshold Value for Biased Random Selection (from 0 to Floor Limit value).
<i>Output Data</i>	None	
<i>Return Code</i>	PP_OK	No error found.
	PP_CRITICAL_MISTAKES	Critical error need to terminate the transaction.
<i>Get Last Error Code</i>	d_PP_LASTERR_MISSING_TERMINAL_DATA	0x0075
	d_PP_LASTERR_SEND_APDU_CMD_FAIL	0x0004
	d_PP_LASTERR_GET_DATA_CMD_ERROR	0x0021

<i>Sample Code</i>	<pre> ULONG ulLastAmount, ulThresholdValue; BYTE bTargetPerc, bMaxTargetPerc, PAN[20]; WORD len; //Get PAN PP_iEMVGetData(0x005A, &len, PAN); *** Get last transaction amount in data pool by the PAN number *** bTargetPerc = 5; bMaxTargetPerc = 20; ulThresholdValue = 100; rtn = PP_iRiskManagement(ulLastAmount, bTargetPerc, bMaxTargetPerc, ulThresholdValue); if (r != PP_OK) { return d_STATUS_CRITICAL_MISTAKES; } </pre>
--------------------	---

4.10 PP_iTermActionAnalysis

This function make the terminal action analysis, and pbTermDecision must return 0x00 (AAC), 0x40 (TC) or 0x80 (ARQC) indicating the result from the analysis.

Syntax C	short PP_iTermActionAnalysis(BYTE bFlags, BYTE *pbTACDenial, BYTE *pbTACOnline, BYTE *pbTermDecision)		
Input Data	bFlags	FLG_MF_ONLINE: The merchant force the transaction online. FLG_EXC_FILE: PAN is in the exception list. FLG_MF_ACCEPT: The merchant force the transaction accepted.	
	pbTACDenial	Terminal Action Code - Denial (5 bytes).	
	pbTACOnline	Terminal Action Code - Online (5 bytes).	
		pbTermDecision	Indicating the result from the analysis (1 byte).
Output Data		0x00	AAC
		0x40	TC
		0x80	ARQC
Return Code	PP_OK	No error found.	
Get Last Error Code	Null		

<i>Sample Code</i>	<pre> BYTE bFlags, CID, TACDenial[5], TACOnline[5]; memcpy(TACDenial, "\x00\x00\x00\x00\x00", 5); memcpy(TACOnline, "\x00\x00\x00\x00\x00", 5); bFlags = 0; if (isExceptionList == TRUE) bFlags = FLG_EXEC_FILE; if (isOnlineForce == TRUE) bFlags = FLG_MF_ONLINE; if (isOnlineForce == TRUE) bFlags = FLG_MF_ACCEPT; PP_iTermActionAnalysis(bFlags, TACDenial, TACOnline, &CID); </pre>
--------------------	--

4.11 PP_iCardActionAnalysis

This function does the card action analysis (1st Generate AC) so that **bTermDecision** is the value returned **PP_iTermActionAnalysis**.

<i>Syntax C</i>	short PP_iCardActionAnalysis(BYTE bTermDecision, BYTE *pbDefTDOL, WORD uiDefTDOLLen, BYTE *pbCryptInfoData)	
<i>Input Data</i>	bTermDecision The value returned PP_iTermActionAnalysis. pbDefTDOL Pointer to the "default terminal TDOL". uiDefTDOLLen Size of "default TDOL" in bytes. In case this information doesn't exist, this value should be 0 (zero).	
<i>Output Data</i>	pbCryptInfoData	<i>Cryptogram Information Data</i> (tag '9F27') , indicating the result from the analysis (1 byte).
<i>Return Code</i>	PP_OK PP_CRITICAL_MISTAKES PP_GAC1_6985_FALLBACK PP_FORCE_ACCEPTANCE	No error found. Critical error need to terminate the transaction.
<i>Get Last Error Code</i>	d_PP_LASTERR_SEND_APDU_CMD_FAIL d_PP_LASTERR_CAS_GAC1_FAIL d_PP_LASTERR_CAS_GAC1_6985_FALLBACK d_PP_LASTERR_IAP_BAD_PADDING d_PP_LASTERR_DIS_GAC_UNKNOW_TAG d_PP_LASTERR_DIS_GAC_ATC_LEN_WRONG d_PP_LASTERR_DIS_GAC_DATA_MISSING d_PP_LASTERR_DIS_GAC_AC_LEN_WRONG d_PP_LASTERR_DIS_GAC_CID_LEN_WRONG	0x0004 0x0023 0x007F 0x0081 0x0072 0x0083 0x0084 0x0086 0x0087

	d_PP_LASTERR_CAS_GAC1_FAIL 0x0023 d_PP_LASTERR_CAS_MERCHANT_FORCE_APPROVED 0x0024 d_PP_LASTERR_CAS_OFFLINE_APPROVED 0x00C8 d_PP_LASTERR_CAS_OFFLINE_DECLINED 0x00C9 d_PP_LASTERR_CAS_NEED_ONLINE 0x0027 d_PP_LASTERR_CAS_CARD_INIT_REFERRAL 0x0028 d_PP_LASTERR_CAS_DDAAC_FAIL 0x0029 d_PP_LASTERR_CAS_DDAAC_FAIL_NEED_GAC2 0x007B d_PP_LASTERR_CAS_ONLINE_FAIL_APPROVAL 0x002A d_PP_LASTERR_CAS_ONLINE_FAIL_DECLINED 0x0078 d_PP_LASTERR_CAS_GAC1_6985_FALLBACK 0x007F d_PP_LASTERR_CAS_GAC1_WRONG_CID 0x002C d_PP_LASTERR_CAS_GAC1_FORMAT_1_PADDING 0x0036 d_PP_LASTERR_CAS_ONLINE_APPROVED 0x00CA d_PP_LASTERR_CAS_ONLINE_DECLINED 0x00CB d_PP_LASTERR_CAS_ONLINE_DECLINED_REVERSAL 0x00CC d_PP_LASTERR_CAS_REFERRAL_APPROVED 0x00CD d_PP_LASTERR_CAS_REFERRAL_DECLINED 0x00CE d_PP_LASTERR_CAS_GAC2_FAIL 0x002E d_PP_LASTERR_DDAAC_DATA_ERROR 0x0040 d_PP_LASTERR_DDAAC_ALGORITHM_NOT_SUPPORT 0x0041 d_PP_LASTERR_DDAAC_HASH1_ERROR 0x0045 d_PP_LASTERR_DDAAC_HASH2_ERROR 0x0046
<i>Sample Code</i>	<pre> BYTE CID, DefaultTDOL[20]; PP_iTermActionAnalysis(bFlags, TACDenial, TACOnline, &CID); memcpy(DefaultTDOL, "\x9F\x27\x02", 3); rtn = PP_iCardActionAnalysis(CID, DefaultTDOL, 3, &CID); if (r != PP_OK) return d_ STATUS_CRITICAL_MISTAKES; if (CID == 0x00) printf("Offline Declined!"); else if (CID == 0x40) printf("Offline Approval!"); else if (CID == 0x80) printf("Online Processing..."); </pre>

4.12 PP_iDefTermActionAnalysis

This function is called only in the case of a communication fail with the host, to attend the "unable to go online" procedure. In this case, **pbTermDecision** will return only 0x00 (AAC) or 0x40 (TC) indicating the result from the analysis.

<i>Syntax C</i>	short PP_iDefTermActionAnalysis(BYTE *pbTACDefault, BYTE *pbTermDecision)	
<i>Input Data</i>	pbTACDefault	Terminal Action Code - Default (5 bytes).
<i>Output Data</i>	pbTermDecision	The result from the analysis 0x40 TC 0x00 AAC
<i>Return Code</i>	PP_OK	No error found.
<i>Get Last Error Code</i>	Null	
<i>Sample Code</i>	BYTE TACDefault[5], bTermDecision;; memcpy(TACDefault, "\x00\x00\x00\x00\x00", 5); PP_iDefTermActionAnalysis (TACDefault, &bTermDecision);	

4.13 PP_iCompletion

This function should be called at the end of the processing, in case **PP_iActionAnalysis** has indicated authorization online, independently of the result of that authorization.

PP_iCompletion is responsible for the following steps of the treatment of the EMV card.

Issuer Authentication;

Script Processing; and

Completion.

At the end of **PP_iCompletion**, the final decision is returned informing to deny or to authorize the transaction.

<i>Syntax C</i>	short PP_iCompletion(CompleteFuncPara *para) typedef struct { short iAction; BYTE bTermDecision; char *pcAuthRespCode; BYTE *pblssAutData; WORD uilssAuthDataLen;
-----------------	---

	<pre> BYTE *pbScript; WORD uiScriptLen; BYTE *pbDefTDOL; WORD uiDefTDOLLen; BYTE bCryptInfoData; WORD uiScriptResLen; BYTE *pbScriptRes; } CompleteFuncPara; </pre>	
<i>Input Data</i>	iAction	<p>Action to be considered by the processing:</p> <p>ACT_ONL_APPR Transaction approved by the host.</p> <p>ACT_ONL_DENY Transaction declined by the host.</p> <p>ACT_UNAB_ONL It was not possible to connect the host.</p> <p>ACT_ONL_ISSUER_REFERRAL_APPR Issuer referral decides to approval this transaction.</p> <p>ACT_ONL_ISSUER_REFERRAL_DENY Issuer referral decides to decline this transaction.</p>
	bTermDecision	This indicate what action must be passed in the 2nd Generate AC (0x00 or 0x40).
	pcAuthRespCode	Host authorization Code ("Authorization Response Code" = Tag '8A'), must be 2 ASCII chars.
	pblssAutData	Pointer for the emitter authentication data ("Issuer Authentication Data" - Tag '91'), only data (no TL).
	uilssAuthDataLen	Size of "Issuer Authentication Data" (from 8 to 16).
	pbScript	Pointer for the emitter scripts, composed by one or more '71' or '72' concatenated templates (complete TLV structures).
	uiScriptLen	Scripts size.
	pbDefTDOL	Pointer for "default TDOL" of the terminal.
	uiDefTDOLLen	Amount of bytes of "default." TDOL In case this information doesn't exist, this value should be 0 (zero).
<i>Output Data</i>	bCryptInfoData	<i>Cryptogram Information Data</i> (tag '9F27') , indicating the final result from the analysis (1 byte).
	uiScriptResLen	Size of obtained <i>Issuer Script Results</i> .

	pbScriptRes	Obtained <i>Issuer Script Results</i> .
<i>Return Code</i>	PP_OK	No error found.
	PP_CRITICAL_MISTAKES	Critical error need to terminate the transaction
	PP_FORCE_ACCEPTANCE	.
<i>Get Last Error Code</i>	d_PP_LASTERR_SEND_APDU_CMD_FAIL	0x0004
	d_PP_LASTERR_CAS_MERCHANT_FORCE_APPROVED	0x0024
	d_PP_LASTERR_CAS_DDAAC_FAIL	0x0029
	d_PP_LASTERR_CAS_GAC2_FAIL	0x002E
	d_PP_LASTERR_DIS_ADF_UNKNOW_TAG	0x002F
	d_PP_LASTERR_DIS_ADF_DATA_MISSING	0x0030
	d_PP_LASTERR_DIS_FCI_DATA_ERROR	0x0031
	d_PP_LASTERR_DIS_TLV_TAG_ZERO	0x0032
	d_PP_LASTERR_DIS_TLV_FAIL	0x0033
	d_PP_LASTERR_DIS_TLV_EXCEED_MAX_LEN	0x0035
	d_PP_LASTERR_IAP_BAD_PADDING	0x0081
	d_PP_LASTERR_DDAAC_DATA_ERROR	0x0040
	d_PP_LASTERR_DDAAC_ALGORITHM_NOT_SUPPORT	0x0041
	d_PP_LASTERR_DDAAC_HASH1_ERROR	0x0045
	d_PP_LASTERR_DDAAC_HASH2_ERROR	0x0046
	d_PP_LASTERR_MISSING_TERMINAL_DATA	0x0075
	d_PP_LASTERR_CARD_DATA_BUF_OVERFLOW	0x0077
	d_PP_LASTERR_DIS_GAC_DATA_MISSING	0x0084
	d_PP_LASTERR_CAS_ONLINE_APPROVED	0x00CA
	d_PP_LASTERR_CAS_ONLINE_DECLINED	0x00CB
	d_PP_LASTERR_CAS_ONLINE_DECLINED_REVERSAL	0x00CC
	d_PP_LASTERR_CAS_REFERRAL_APPROVED	0x00CD
	d_PP_LASTERR_CAS_REFERRAL_DECLINED	0x00CE
	d_PP_LASTERR_CAS_ONLINE_FAIL_APPROVAL	0x002A
	d_PP_LASTERR_CAS_ONLINE_FAIL_DECLINED	0x0078

<p><i>Sample Code</i></p>	<pre> CompleteFuncPara stCompletePara; BYTE ARC[2], IAD[10], Script[100], bScriptRes[100]; memcpy(ARC, "\x30\x30", 2); memcpy(IAD, "1234567890", 10); memcpy(Script, "\x71\x18\x9F\x18\x04\x00\x00\x00\x01\x86\x0F\x8C\x24 \x00\x00\x0A\x8E\x08\x11\x22\x33\x44\x55\x66\x77\x88", 26); stCompletePara.iAction = ACT_ONL_APPR; stCompletePara.bTermDecision = 0x40; stCompletePara.pcAuthRespCode = ARC; stCompletePara.uiLssAuthDataLen = 10; stCompletePara.pblssAutData = IAD; stCompletePara.uiScriptLen = 26; stCompletePara.pbScript = Script; stCompletePara.uiDefTDOLLen = 0; stCompletePara.pbDefTDOL = NULL; stCompletePara.pbScriptRes = bScriptRes; rtn = PP_iCompletion(&stCompletePara); if (rtn == PP_OK) { if (stCompletePara.bCryptInfoData == 0x40) printf("Online Approval"); else if (stCompletePara.bCryptInfoData == 0x00) printf("Online Declined"); } </pre>
---------------------------	---

4.14 PP_iEMVGetData

This function allows the extraction of resulting data of the processing of cards EMV, could be original of the card (like PAN, for instance) or of the processing of the kernel (like TVR, TSI, CVM Results, etc.).

It can be called by the main application at any moment after **PP_iSelectApp**.

<i>Syntax C</i>	short PP_iEMVGetData(WORD wTag, WORD *puiLen, BYTE *pbVal)	
<i>Input Data</i>	wTag	Tag of the data
<i>Output Data</i>	puiLen	Return data length
	pbVal	Return data
<i>Return Code</i>	PP_OK	No error found.
	PP_DATA_NOT_FOUND	Data not found.
<i>Get Last Error Code</i>	Null	
<i>Sample Code</i>	BYTE TVR[5]; WORD len; PP_iEMVGetData(0x0095, TVR, &len);	

4.15 PP_iGetLastError

Get detail internal information about the last error that occurred in the above PP_iXXX functions.

Note that, for the returning code of PP_iGetLastError function, it cannot be used to identify if the transaction should be continued or aborted. This is because in some cases the PP_iXXX function returns PP_OK, however, the returning code of PP_iGetLastError function may not be 0 (0 means no error). The reason why the EMV kernel still allows the transaction to be continued is because it is not to abort the transaction according to EMV specification. This returning code just records the minor status.

<i>Syntax C</i>	short PP_iGetLastError(void)	
<i>Input Data</i>	None	
<i>Output Data</i>	None	
<i>Return Code</i>	PP_OK	No error found.
	Other	Refer the last error code list table.(6.7)

<i>Sample Code</i>	<pre> USHORT usLastError; usLastError = PP_iGetLastError(); if (usLastError != PP_OK) { switch(usLastError) { case d_PP_LASTERR_FUNCTION_NOT_SUPPORTED: printf("Function not supported!"); break; . . . } } </pre>
--------------------	---

5. Functions to Be Supplied by the Application

The functions described in this item should be supplied by the main application, in way that they can be called by the kernel EMV when necessary.

5.1 APP_fGetCAPK

This function should be supplied by the main application and it is called by the kernel EMV during the processing, in a way to obtain the public key of the certificatory entity used in the authentication of the card (SDA or DDA).

<i>Syntax C</i>	BYTE APP_fGetCAPK(BYTE *pbRID, BYTE bCAPKIdx, BYTE *pbMod, WORD *puiModLen, BYTE *pbExp, WORD *puiExpLen);	
<i>Input Data</i>	pbRID	RID of the application, appearing for 5 bytes.
	bCAPKIdx	Index of the key for supplied RID (Certification Authority Public Key Index).
<i>Output Data</i>	pbMod	Module of obtained key.
	puiModLen	Size of the module, in bytes (up to 248).
	pbExp	Exponent of the key.
	puiExpLen	Size of the exponent (1 or 3).
<i>Return Code</i>	TRUE	Key found.
	FALSE	Key not found.

5.2 APP_fCheckRevocation

This function should be supplied by the main application and it is called by the EMV kernel during the processing of the authentication, in a way to verify a certain certificate is revoked.

<i>Syntax C</i>	BYTE APP_fCheckRevocation(BYTE *pbRID, BYTE bCAPKIdx, BYTE *pbSerNum)	
<i>Input Data</i>	pbRID	RID of the application, pointing to 5 bytes.
	bCAPKIdx	Index of the key for RID supplied (Certification Authority Public Key Index).
	pbSerNum	Certificate Serial Number (3 bytes).
<i>Output Data</i>	None	
<i>Return Code</i>	TRUE	Revocation found.
	FALSE	Revocation not found.
<i>Sample Code</i>		

5.3 APP_fGetTermData

This function allows to the kernel to obtain data of the terminal whenever necessary to the processing of cards EMV (as the Terminal Country Code, for instance)..

<i>Syntax C</i>	BYTE APP_fGetTermData(WORD wTag, WORD *puiLen, BYTE *pbVal)	
<i>Input Data</i>	wTag	Data identification "Tag" .
<i>Output Data</i>	puiLen	Size of Data Obtained (bytes).
	pbVal	Data obtained.
<i>Return Code</i>	TRUE	Data found.
	FALSE	Data not found.
<i>Sample Code</i>		

5.4 APP_iGetPINOffline

This function can be used by the kernel during the processing of **EMV_iCardholderVerification** for the capture of Plain text PIN so the card can verify it offline (if it will be send in plain Text or Encrypted will be done by the Kernel depending on the card need).

The function **APP_PINVerifResult** is called by the kernel after the verification so that the application can take actions (how to present messages), If necessary.

<i>Syntax C</i>	short APP_iGetPINOffline(short iRemainingTries, char *pszPIN)	
<i>Input Data</i>	iRemainingTries	Number of remaining attempts for the presentation of PIN (if not known, this value is zero).
<i>Output Data</i>	pszPIN	Typed PIN (Plain Text), with NUL terminator,
<i>Return Code</i>	PP_OK PP_PIN_BY_PASS PP_CRITICAL_MISTAKES	PIN input ok. By pass the PIN input. Critical mistake found, must terminate the transaction.
<i>Sample Code</i>		

5.5 APP_iGetPINOnline

This function can be used by the kernel during the processing of **PP_iCardholderVerification** for the capture of PIN of the card for *online* verification.

The main application is responsible for obtaining PIN and cryptographs it according to the net acquirer's specific algorithm, and no data is returned to the kernel. In case the operation is OK, the Encrypted PIN is stored by the application for subsequent sending to Host.

<i>Syntax C</i>	short APP_iGetPINOnline(void)	
<i>Input Data</i>	None	
<i>Output Data</i>	None	
<i>Return Code</i>	PP_OK PP_PIN_BY_PASS PP_CRITICAL_MISTAKE S	PIN input ok. By pass the PIN input. Critical mistake found, must terminate the transaction.
<i>Sample Code</i>		

5.6 APP_PINVerifyResult

This function can be used by the kernel after the *offline* PIN verification so that the application can take actions (how to present messages), If necessary.

<i>Syntax C</i>	void APP_PINVerifyResult(short iResult)	
<i>Input Data</i>	iResult	Result of the verification: PINRES_OK PIN validated with success.

		PINRES_FAIL PIN doesn't check. PINRES_BLOCKED The PIN was already blocked. PINRES_FAILBLOCKED The last PIN Entry was invalid and the card was blocked
<i>Output Data</i>	None	
<i>Return Code</i>	None	
<i>Sample Code</i>	<pre> void APP_PINVerifyResult(short iResult) { if (iResult == PINRES_OK) { print("\nPIN OK!"); } else if (iResult == PINRES_FAIL) { print("\nPIN FAIL!"); } else if (iResult == PINRES_BLOCKED) { print("\nPIN BLOCKED!"); } } </pre>	

5.7 APP_ShowAdvice

Show advice message.

<i>Syntax C</i>	void APP_ShowAdvice(BYTE advice)	
<i>Input Data</i>	advice	This value is form Cryptogram Information Data (CID) bit 3-1. bit 3-1 (Reason/Advice/Referral Code) 000 No information given 001 Service not allowed 010 PIN Try Limit exceeded 011 Issuer authentication failed xxx All other value are RFU Ps: For more detail about CID, please refer section 6.5.5.4 of EMV 4.3 BOOK 3.
<i>Output Data</i>	None	

<i>Return Code</i>	None
<i>Sample Code</i>	<pre> void APP_ShowAdvice(BYTE advice) { if(advice & 0x08) { //Advice required advice &= 0x07; switch(advice) { case 1: //Service not allowed break; case 2: //PIN Try Limit exceeded break; } } } </pre>

5.8 APP_iIDCardVerify

Display the cardholder's ID card information for merchant to verify. This CVM only be used in China market (for PBOC)

<i>Syntax C</i>	short APP_iIDCardVerify(void)	
<i>Input Data</i>	None	
<i>Output Data</i>	None	
<i>Return Code</i>	PP_OK <> PP_OK	ID Card verified ok ID Card verified failed
<i>Sample Code</i>	<pre> short APP_iIDCardVerify() { BYTE id_type, id_num[41], str[17]; WORD usTagLen; wub_memset(id_num, 0x00, sizeof(id_num)); </pre>	

```
str[16] = 0;
```

```
VirtualScreenClear();
```

```
VirtualScreenPrint((BYTE*)"** ID Verify **", FALSE);
```

```
if(PP_iEMVGetData(0x9F62, &usTagLen, &id_type) ==  
PP_DATA_NOT_FOUND)
```

```
{  
    return 1;  
}
```

```
if(PP_iEMVGetData(0x9F61, &usTagLen, id_num) ==  
PP_DATA_NOT_FOUND)
```

```
{  
    return 1;  
}
```

```
VirtualScreenPrint((BYTE*)"ID Type : ", TRUE);
```

```
if (id_type == 0x00)
```

```
    VirtualScreenPrint((BYTE*)" -> ID Card", TRUE);
```

```
else if (id_type == 0x01)
```

```
    VirtualScreenPrint((BYTE*)" -> Military", TRUE);
```

```
else if (id_type == 0x02)
```

```
    VirtualScreenPrint((BYTE*)" -> Passport", TRUE);
```

```
else if (id_type == 0x03)
```

```
    VirtualScreenPrint((BYTE*)" -> Immigration", TRUE);
```

```
else if (id_type == 0x04)
```

```
    VirtualScreenPrint((BYTE*)" -> Temp ID Card", TRUE);
```

```
else if (id_type == 0x05)
```

```
    VirtualScreenPrint((BYTE*)" -> Others", TRUE);
```

```
VirtualScreenPrint((BYTE*)"ID Number : ", TRUE);
```

```
VirtualScreenPrint(id_num, TRUE);
```

```
VirtualScreenPrint((BYTE*)" [ ENTER for OK ]", TRUE);
```

	<pre>if (VirtualScreenDisplay() == TRUE) return 0; //Verify OK return 1; }</pre>
--	---

6. Appendix. Data Definition

6.1 Return Code

Definition	Value
PP_OK	0
PP_DATA_BUFFER_EXCEEDED	1
PP_INVALID_PARA	2
PP_CRITICAL_MISTAKES	3
PP_SELECTION_FAIL	4
PP_PIN_BY_PASS	5
PP_TERMINAL_DATA_MISSING	6
PP_DATA_NOT_FOUND	7
PP_GAC1_6985_FALLBACK	8
PP_ONLY_1_AP_NO_FALLBACK	9
PP_IAP_9481_FALLBACK	10
PP_FORCE_ACCEPTANCE	11
PP_FUNCTION_NOT_SUPPORTED	12
PP_RETURN_KMS_OFFLINE_ENC_PIN	13

6.2 AUC Flags

Definition	Value
USCTRL_CASHBACK	0x01
USCTRL_SERVICES	0x02
USCTRL_GOODS	0x04
USCTRL_ATM	0x08

6.3 TAA Flags

Definition	Value
FLG_MF_ONLINE	0x01
FLG_EXEC_FILE	0x02
FLG_MF_ACCEPT	0x04

6.4 Online Flags

Definition	Value
ACT_ONL_APPR	1
ACT_ONL_DENY	2
ACT_UNAB_ONL	3
ACT_ONL_ISSUER_REFERRAL_APPR	4
ACT_ONL_ISSUER_REFERRAL_DENY	5

6.5 PIN Verify Display Flags

Definition	Value
PINRES_OK	1
PINRES_FAIL	2
PINRES_BLOCKED	3
PINRES_FAILBLOCKED	4

6.6 Last Error Codes

Error list by PP_iGetLastError

Code	Value	Description
For EMV Library Error		
d_PP_LASTERR_FUNCTION_NOT_SUPPORTED	0x0001	The responded data of APDU is 0x6A81(SW12).
d_PP_LASTERR_CANDIDATE_LIST_FULL	0x0002	Candidate list full.
d_PP_LASTERR_SEND_APDU_CMD_FAIL	0x0004	Send APDU command is failure.
d_PP_LASTERR_ONLY_1_AP_BLOCKED	0x0026	Only one application blocked
d_PP_LASTERR_APPLICATION_NOT_ALLOW	0x0007	Select application file error.
d_PP_LASTERR_IAP_AIP_AFL_ERROR	0x0008	Dismantle AIP and AFL error.
d_PP_LASTERR_IAP_UNKNOW_SW12	0x0009	Card APDU response unknown.
d_PP_LASTERR_IAP_MISS_PAN	0x000A	The Primary Account Number is lost.
d_PP_LASTERR_IAP_MISS_CDOL1	0x000B	The Card Risk Management Data Object List 1 is lost.
d_PP_LASTERR_IAP_MISS_CDOL2	0x000C	The Card Risk Management Data Object List 2 is lost.
d_PP_LASTERR_IAP_APPLICATION_NOT_ALLOW	0x000D	Card APDU response 0x6985(SW12).
d_PP_LASTERR_IAP_BAD_PADDING	0x0081	The responded data of APDU is wrong.
d_PP_LASTERR_IAP_FALLBACK_9481	0x0088	The responded data of APDU is 0x9481(SW12).
d_PP_LASTERR_ERROR_9F4A_RULE	0x000E	The static data authentication tag list does meet with the rules.
d_PP_LASTERR_KEY_NO_FOUND	0x0010	Not found key.
d_PP_LASTERR_PAN_NOT_SAME	0x0011	The PAN isn't the same.
d_PP_LASTERR_DDOL_MISS	0x0012	The DDOL is lost.
d_PP_LASTERR_INTERNAL_AUTHENTICATE_FAIL	0x0013	Internal authenticate fail.
d_PP_LASTERR_SDA_LENGTH_NOT_MATCH	0x0014	The SDA length isn't match.
d_PP_LASTERR_NO_OFFLINE_DATA_AUTH_MATCH	0x0015	Offline data authentication data isn't match.
d_PP_LASTERR_READ_DATA_TAG_NOT_70	0x0016	The record not TVR format , fail offline data authenticate.
d_PP_LASTERR_MISS_APP_EXPIRATION_DATE	0x0017	The application effective date is lost
d_PP_LASTERR_CARDHOLDER_VER_NOT_SUPP	0x0018	Cardholder verification isn't supported.
d_PP_LASTERR_CVM_LIST_MISSING	0x0019	CVM list is lost.
d_PP_LASTERR_CVM_FINISH	0x001A	CVM finish.
d_PP_LASTERR_CVM_PLAIN_TEXT	0x001B	CVM plain text
d_PP_LASTERR_CVM_ENC_PIN_ONLINE	0x001C	CVM enciphered PIN verified online.
d_PP_LASTERR_CVM_PLAINTEXT_PIN_SIGNATURE	0x001D	CVM plaintext PIN signature
d_PP_LASTERR_CVM_ENC_PIN	0x001E	CVM enciphered PIN
d_PP_LASTERR_CVM_ENC_PIN_SIGNATURE_OFFLINE	0x001F	CVM enciphered PIN verified offline
d_PP_LASTERR_CVM_SIGNATURE	0x0020	CVM signature
d_PP_LASTERR_CVM_IDCARD_VERIFY	0x0025	CVM cardhold verify
d_PP_LASTERR_GET_DATA_CMD_ERROR	0x0021	Get data command error.
d_PP_LASTERR_CAS_GAC1_FAIL	0x0023	Generate AC1 fail.
d_PP_LASTERR_CAS_MERCHANT_FORCE_APPROVED	0x0024	Merchant force approved.
d_PP_LASTERR_CAS_OFFLINE_APPROVED	0x00C8	Offline approved of PP_iCardActionAnalysis().
d_PP_LASTERR_CAS_OFFLINE_DECLINED	0x00C9	Offline declined of PP_iCardActionAnalysis().
d_PP_LASTERR_CAS_NEED_ONLINE	0x0027	Offline need online of PP_iCardActionAnalysis().
d_PP_LASTERR_CAS_CARD_INIT_REFERRAL	0x0028	Init referral of PP_iCardActionAnalysis().
d_PP_LASTERR_CAS_DDAAC_FAIL	0x0029	Dynamic data authentication AC fail.
d_PP_LASTERR_CAS_DDAAC_FAIL_NEED_GAC2	0x007B	Dynamic data authentication AC fail but need generate AC2.
d_PP_LASTERR_CAS_ONLINE_FAIL_APPROVAL	0x002A	Online fail approval
d_PP_LASTERR_CAS_ONLINE_FAIL_DECLINED	0x0078	Online fail declined
d_PP_LASTERR_CAS_GAC1_6985_FALLBACK	0x007F	The responded data of APDU is 0x6985(SW12).
d_PP_LASTERR_CAS_GAC1_WRONG_CID	0x002C	Card return CID is wrong when generate AC1.
d_PP_LASTERR_CAS_GAC1_FORMAT_1_PADDING	0x0036	Generate AC1 format is fail.
d_PP_LASTERR_CAS_ONLINE_APPROVED	0x00CA	Online approved of PP_iCompletion() or PP_iCardActionAnalysis().
d_PP_LASTERR_CAS_ONLINE_DECLINED	0x00CB	Online declined of PP_iCompletion() or PP_iCardActionAnalysis().
d_PP_LASTERR_CAS_ONLINE_DECLINED_REVERSAL	0x00CC	Online declined reversal of PP_iCompletion() or PP_iCardActionAnalysis().
d_PP_LASTERR_CAS_REFERRAL_APPROVED	0x00CD	Referral approved
d_PP_LASTERR_CAS_REFERRAL_DECLINED	0x00CE	Referral declined
d_PP_LASTERR_CAS_GAC2_FAIL	0x002E	Generate AC2 fail.
d_PP_LASTERR_DIS_ADF_UNKNOW_TAG	0x002F	Dismantle ADF has unknown tag.
d_PP_LASTERR_DIS_ADF_DATA_MISSING	0x0030	Miss the ADF data.
d_PP_LASTERR_DIS_FCI_DATA_ERROR	0x0031	FCI data error when dismantling.
d_PP_LASTERR_DIS_TLV_TAG_ZERO	0x0032	TLV tag second byte is zero.
d_PP_LASTERR_DIS_TLV_FAIL	0x0033	TLV error when dismantling.
d_PP_LASTERR_DIS_TLV_EXCEED_MAX_LEN	0x0035	Tag value length over the total length.
d_PP_LASTERR_MISS_AIP_AFL	0x0037	Miss the AIP and AFL.

d_PP_LASTERR_DDOL_NOT_HAVE_9F37	0x0038	DDOL have not tag 9F37.
d_PP_LASTERR_DGP_DIS_TLV_FAIL	0x0039	Dismantle TLV fail when PP_iSelect_App().
d_PP_LASTERR_MULTIPLE_OCCURENCE	0x003A	TLV list multiple occurrence when PP_iSelect_App().
d_PP_LASTERR_CVM_DATA_LEN_LESS_8	0x003B	The CVM data length less 8.
d_PP_LASTERR_SFI_TEMPLATE_RULE_WRONG	0x007C	SFI above 10 Proprietary data, tag = 0x70 is not necessary
d_PP_LASTERR_SDA_DATA_ERROR	0x003C	Static Data Authentication data is error.
d_PP_LASTERR_SDA_ALGORITHM_NOT_SUPPORT	0x003D	Static Data Authentication algorithm not support.
d_PP_LASTERR_DDA_DATA_ERROR	0x003E	Dynamic Data Authentication data is error.
d_PP_LASTERR_DDAAC_DATA_ERROR	0x0040	Dynamic Data Authentication AC data is error.
d_PP_LASTERR_DDAAC_ALGORITHM_NOT_SUPPORT	0x0041	Dynamic Data Authentication algorithm not support.
d_PP_LASTERR_KEY_LENGTH_ERROR	0x0043	RFU
d_PP_LASTERR_DDAAC_HASH1_ERROR	0x0045	Hash1 fail when Dynamic Data Authentication AC.
d_PP_LASTERR_DDAAC_HASH2_ERROR	0x0046	Hash2 fail when Dynamic Data Authentication AC.
d_PP_LASTERR_ISSUER_CERT_NOT_EXIST	0x0047	Issuer Public Key Certificate not exist.
d_PP_LASTERR_ISSUER_CERT_FORMAT_ERROR	0x0048	Issuer Public Key Certificate format is error.
d_PP_LASTERR_ISSUER_CERT_IIN_PAN_NOT_SAME	0x0049	Issuer identification number and PAN not same.
d_PP_LASTERR_ISSUER_CERT_REVOCATION_FOUND	0x004A	Issuer Public Key Certificate revocation found
d_PP_LASTERR_ISSUER_CERT_ALGORITHM_NOT_SUPPORT	0x004B	Issuer Public Key Certificate algorithm not support.
d_PP_LASTERR_ISSUER_CERT_LENGTH_ERROR	0x004C	Issuer Public Key Certificate length is error.
d_PP_LASTERR_ISSUER_CERT_EXPIRATION_DATE	0x004D	Issuer Public Key Certificate has expired.
d_PP_LASTERR_ISSUER_CERT_HASH_NOT_MATCH	0x004E	Issuer Public Key Certificate hash is not mach.
d_PP_LASTERR_ISSUER_CERT_EXPONENT_NOT_EXIST	0x004F	Issuer Public Key Certificate exponent is not exist.
d_PP_LASTERR_ISSUER_CERT_REMAINDER_MISSING	0x0050	Issuer Public Key Certificate pomander is lost.
d_PP_LASTERR_ISSUER_CERT_CAPKI_NOT_EXIST	0x0079	Issuer Public Key Certificate CAPKI is not exist.
d_PP_LASTERR_ICC_CERT_NOT_EXIST	0x0051	ICC Public Key Certificate not exist.
d_PP_LASTERR_ICC_CERT_FORMAT_ERROR	0x0052	ICC Public Key Certificate format error
d_PP_LASTERR_ICC_CERT_ALGORITHM_NOT_SUPPORT	0x0053	ICC Public Key Certificate algorithm not support.
d_PP_LASTERR_ICC_CERT_LENGTH_ERROR	0x0054	ICC Public Key Certificate length is error.
d_PP_LASTERR_ICC_CERT_HASH_NOT_MATCH	0x0055	ICC Public Key Certificate hash is not mach.
d_PP_LASTERR_ICC_CERT_EXPIRATION_DATE	0x0056	ICC Public Key Certificate has expired.
d_PP_LASTERR_ICC_CERT_EXPONENT_NOT_EXIST	0x0057	ICC Public Key Certificate exponent is not exist.
d_PP_LASTERR_ICC_ISSUER_PK_NOT_EXIST	0x0058	ICC Public Key Certificate PK is not exist.
d_PP_LASTERR_ICC_CERT_REMAINDER_MISSING	0x0059	ICC Public Key Certificate pomander is lost.
d_PP_LASTERR_PIN_CERT_NOT_EXIST	0x005A	PIN Public Key Certificate is not exit.
d_PP_LASTERR_PIN_CERT_LENGTH_ERROR	0x005B	PIN Public Key Certificate length is not equal to CAPK Modulus
d_PP_LASTERR_PIN_CERT_FORMAT_ERROR	0x005C	PIN Public Key Certificate format is error.
d_PP_LASTERR_PIN_CERT_ALGORITHM_NOT_SUPPORT	0x005D	PIN Public Key Certificate algorithm not support.
d_PP_LASTERR_PIN_CERT_HASH_NOT_MATCH	0x005E	PIN Public Key Certificate hash is not mach.
d_PP_LASTERR_PIN_CERT_EXPIRATION_DATE	0x005F	PIN Public Key Certificate has expired.
d_PP_LASTERR_PIN_CERT_KEY_LENGTH_ERROR	0x0060	PIN Public Key length is error.
d_PP_LASTERR_PIN_CERT_EXP_NOT_EXIST	0x0085	PIN Encipherment Public Key exponent is not exist.
d_PP_LASTERR_READ_PIN_TRY_COUNT_FAIL	0x0061	Card APDU response is fail when get PIN try counter.
d_PP_LASTERR_CVM_PLAIN_TEXT_PIN_NOT_KEYIN	0x0062	Plaintext PIN verification performed by ICC but cardholder didn't input the PIN.
d_PP_LASTERR_CVM_PLAIN_TEXT_PIN_TRY_LIMIT_EXCEEDED	0x0063	PIN Try Limit exceeded.
d_PP_LASTERR_CVM_PLAIN_TEXT_PIN_OK	0x0064	Input plaintext PIN verification has successfully
d_PP_LASTERR_CVM_PLAIN_TEXT_PIN_WRONG	0x0066	Input plaintext PIN verification has wrong.
d_PP_LASTERR_CVM_PLAIN_TEXT_UNKNOW_SW12	0x0067	The responded data of APDU is unknown when input plaintext PIN verification.
d_PP_LASTERR_CVM_PLAIN_TEXT_PIN_BLOCK	0x007D	The ICC has been blocked.
d_PP_LASTERR_CVM_TERMINAL_NOT_SUPPORT_SPECIFY_CVM	0x0068	The terminal isn't support specify CVM.
d_PP_LASTERR_CVM_ENC_PIN_ONLINE_PIN_NOT_KEYIN	0x0069	Encipher PIN online verification performed by ICC but cardholder didn't input the PIN.
d_PP_LASTERR_CVM_ENCIPHERED_PIN_NOT_KEYIN	0x006C	Encipher PIN verification performed by ICC but cardholder didn't input the PIN.
d_PP_LASTERR_CVM_ENCIPHERED_PIN_TRY_LIMIT_EXCEEDED	0x006D	Encipher PIN verification performed by ICC but cardholder didn't input the PIN.
d_PP_LASTERR_CVM_ENCIPHERED_PIN_UNKNOW_SW12	0x006E	The responded data of APDU is unknown when input Encipher PIN verification.
d_PP_LASTERR_CVM_ENCIPHERED_PIN_GET_RN_UNKNOW_SW12	0x0071	The responded data of APDU is unknown when send "GET CHALLENGE" command by input Encipher PIN action.
d_PP_LASTERR_MISSING_TERMINAL_DATA	0x0075	Missing terminal data.
d_PP_LASTERR_CARD_DATA_BUF_OVERFLOW	0x0077	Card data buffer overflow.
d_PP_LASTERR_CVM_ENCIPHERED_PIN_OK	0x006F	Input encipher PIN verification has successfully
d_PP_LASTERR_CVM_ENCIPHERED_PIN_WRONG	0x0070	Input encipher PIN verification has wrong.
d_PP_LASTERR_CVM_ENCIPHERED_PIN_BLOCK	0x007E	The ICC has been blocked.

d_PP_LASTERR_DIS_GAC_UNKNOW_TAG	0x0072	Dismantle Application Cryptogram has unknown tag.
d_PP_LASTERR_DIS_GAC_ATC_LEN_WRONG	0x0083	The Application Transaction Counter length is fail when dismantle Application Cryptogram.
d_PP_LASTERR_DIS_GAC_DATA_MISSING	0x0084	Application Cryptogram data is lost.
d_PP_LASTERR_DIS_GAC_AC_LEN_WRONG	0x0086	Application Cryptogram length is fail.
d_PP_LASTERR_DIS_GAC_CID_LEN_WRONG	0x0087	When Dismantle Application Cryptogram CID's length is fail.