PIN_CS04 DLL Specification

Rev 2.5(release)

Henry Huang, Zhang Yin

10/10/2010

Revision History

Ed.	Date	Preparer	Comments
1.0	10/10/2010	Henry Huang	Initial draft
2.0	2/14/2011	Henry Huang	Modified
2.1	8/25/2011	Suzhen Tan	Modified Modified the following chapters 3.7.ZT_PIN_ImportKey 3.8.ZT_PIN_GetPinBlock 3.15.ZT_PIN_WriteKeyValue 3.16.ZT_PIN_ReadEPPInfo 3.19.ZT_PIN_UpdatePassword 3.23.ZT_PIN_ExportRsaKey
2.2	10/10/2011	Huangshangfei	Added 3.28.ZT_PIN_CreateOffset 3.29.ZT_PIN_LocalDes
2.3	30/12/2011	Suzhen Tan	Modified 4.1Error Code List
2.4	18/10/2012	Suzhen Tan	Added 3.30 ZT_PIN_StoreKey
2.5	13/11/2012	Suzhen Tan	Modified 3.15 ZT_PIN_WriteKeyValue

Table of Contents

1. Scope

The purpose of this document is to describe all APIs of the ZT598M/ZT596F/ZT599K EPP, It tells developers how to use the API and the problem should be noted.

2. Software design

A new Release 2.0 specification based on a C API, details as below,

API name: PIN_CS04_xxx.dll

API type: Win32 Dynamic-Link Library

EPP Firmware Version: ZT598_M5x_H12, ZT596_F1x_H13, ZT599_K1x_H20

PC OS: Windows 2000/XP/7

3. APIs List

3.1. ZT_PIN_OpenDevice

Description

Open pin pad device before all other operations.

Prototype

int WINAPI ZT_PIN_OpenDevice(BYTE byType,LPCTSTR szDescription)

Parameters

byType

[in]port type

If using usb communication ,the value is 1

If using serial port communication, the value is 2.

szDescription.

[in]port decription

If using usb communication ,the value is "FT232R USB UART"

If using serial communication, the value is “COMX:115200,N,8,1”

Return Values

If the function succeeds, the return value is zero.

If the function fails, the return value is nonzero. To get extended error information, see *Error Code List*.

3.2. ZT_PIN_CloseDevice

Description

Close pinpad device when all operations are finished.

Prototype

```
int WINAPI ZT_PIN_CloseDevice ( )
```

Parameters

None.

Return Values

If the function succeeds, the return value is zero.

If the function fails, the return value is nonzero. To get extended error information, see *Error Code List*.

Remarks

The application must make sure that "ZT_PIN_OpenDevice" and "ZT_PIN_CloseDevice" will be paired .

3.3. ZT_PIN_Reset

Description

Reset the pinpad device

Prototype

int WINAPI ZT_PIN_Reset ( )

Parameters

None.

Return Values

If the function succeeds, the return value is zero.

If the function fails, the return value is nonzero. To get extended error information, see *Error Code List*.

Remarks

This command is used by an application control program to cause a device to reset itself a known good condition.It does not delete any keys.

3.4. ZT_PIN_DeleteKey

Description

Delete one specified key or one class specified keys that have

been loaded or imported into the Pinpad device .

Prototype

int WINAPI ZT_PIN_DeleteKey(WORD wKeyId)

Parameters

wKeyID
[in] It must be the values as follows, other values is invalid.

if wKeyID=1~599,601~619 the one specifed DES or RSAkey will be erased.
if wKeyID=999, all RSA and DES keys will be erased.

Return Values

If the function succeeds, the return value is zero.

If the function fails, the return value is nonzero. To get extended error information, see *Error Code List*.

3.5. ZT_PIN_SetEntryMode

Description

This command is used to switch pinpad to input mode

Prototype

int WINAPI ZT_PIN_SetEntryMode(BYTE byMode)

Parameters

byMode
[in] It must be the values as follows, other values is invalid.

if byMode=0x30, it means to close the pinpad input.

if byMode=0x31,it means to open pinpad input with pin mode.

it can be used when call WFS_CMD_PIN_GET_PIN

if byMode=0x32, it means to open pinpad input with key mode. this way is reserved

if byMode=0x33,means open pinpad input in clear text mode. it can be used when call WFS_CMD_PIN_GET_DATA

Return Values

If the function succeeds, the return value is zero.

If the function fails, the return value is nonzero. To get extended error information, see *Error Code List*.

3.6. ZT_PIN_ReadInputData

Description

Read the data that have been input by user .

Prototype

```
int WINAPI ZT_PIN_ReadInputData (LPBYTE    lpInputData,
                                 LPBYTE    lpInputDataLen)
```

Parameters

lpInputData
[out] Input data.

lpInputDataLen
[out] Number of input data.

Return Values

If the function succeeds, the return value is zero.

If the function fails, the return value is nonzero. To get extended error information, see *Error Code List*.

Remarks

Under clear text input mode, this command will return actual input data and clear internal buffer.

Under pin input mode, this command will return 0x00 and not clear internal buffer.

Example

WFS_CMD_PIN_GET_DATA function:

```
ZT_PIN_SetEntryMode(0x33); //set clear text input mode
int nNum=0;
BYTE szData[16]={0};
While(nNum <6)
{
    int nDataLen=0;
    ZT_PIN_ReadInputData(szData,& nDataLen); //read input data
     nNum += nDataLen;
}
ZT_PIN_SetEntryMode(0x30); //close input
```

3.7. ZT_PIN_ImportKey

Description

Load the key in the encryption module. the key can be passed in clear text mode or encrypted with an accompanying “key encryption key”.

Prototype

```
int   WINAPI ZT_PIN_ImportKey(WORD  wKeyId,
                              WORD     wEnKeyId,
                              WORD     wVI,
                              LPBYTE   lpValue,
                              BYTE   byValeLen,
                              WORD     wUse,
                              BYTE   byMode,
```

```
                    LPBYTE    lpKCV,
                    LPBYTE    lpKCVL)
```

Parameters

wKeyId

[in] Specifies the index of key being loaded.The value range is 1 to 599.

wEnKeyId

[in] Specifiles a key index which were used to encrypt the key passed in lpValue.

It must be the values as follows, other values is invalid.

If wEnKeyId=1~599, use for “key encryption key”

If wEnKeyId=0xffff, it means no “key encryption key”

wVI

[in] Reserved; must be zero.

lpValue

[in] Specifies the value of key to be loaded.

byValeLen

[in] Specifies the number of bytes to be loaded. The value can be 8, 16 or 24, but only 16 or 24 if use for WFS_PIN_USEKEYENCKEY.

wUse

[in] Specifies the type of access for which the can be used as a combination of the following flags:

Value	Meaning
WFS_PIN_USEKEYENCKEY (0x0020)	Key is used as key encryption key.
WFS_PIN_USEFUNCTION (0x0002)	Key can be used for PIN function.
WFS_PIN_USEMACING (0x0004)	Key can be used for MACing.
WFS_PIN_USECRYPT (0x0001)	Key can be used for encryption/decryption

In general, the type of access must be unique. Only WFS_PIN_USEMACING and WFS_PIN_USECRYPT can be as a combination

byMode
[in] ImportKey mode;the value as below:
0x30:direct stroed
0x31:XOR with existing key

lpKCV
[out] Fixed length 8 bytes; Contains the key verification code data that can be used for verification of the loaded key.

lpKCVL
[out] Specifies the byte number of the *lpKCV*.

Return Values

If the function succeeds, the return value is zero.

If the function fails, the return value is nonzero. To get extended error information, see *Error Code List*.

Remarks

wEnKeyId must be different from wKeyId.

The value of Every key must be different from Others.

3.8. ZT_PIN_GetPinBlock

Description

This function takes the account information and a PIN entered by the user to build a formatted PIN.

Prototype

```
int WINAPI ZT_PIN_GetPinBlock(WORD        wKeyId,
                              WORD        wEnKeyId,
```

```
WORD        wVI,
WORD        wFormat,
LPSTR       lpCustomerData,
LPSTR       lpXORData,
BYTE        byPadding,
BYTE        byPinLen,
LPBYTE      lpPinBlock,
LPBYTE      lpPinBlockLen)
```

Parameters

wKeyId

[in] Specifies the index of key used to encrypt the formatted pin for the first time.The value range is 1 to 599.

wEnKeyId

[in] Reserved; must be zero.

Specifies the index of key used to format the once encrypted formatted PIN, The value range is 1 to 599.0xffff if no second encryption required.

wVI

[in] Reserved; must be zero.

wFormat

[in] Specifies the format of the PIN block ;it can be only have one of the follows values:

Value	Algorithm format
0x0001	WFS_PIN_FORM3624
0x0002	WFS_PIN_FORMANSI
0x0004	WFS_PIN_FORMISO0
0x0008	WFS_PIN_FORMISO1
0x0010	WFS_PIN_FORMECI2
0x0020	WFS_PIN_FORMECI3
0x0040	WFS_PIN_FORMVISA
0x0080	WFS_PIN_FORMDIEBOLD
0x0100	WFS_PIN_FORMDIEBOLDCO
0x0200	WFS_PIN_FORMVISA3
0x0400	WFS_PIN_FORMBANKSYS
0x0800	WFS_PIN_FORMEMV
0x2000	WFS_PIN_FORMISO3

lpCustomerData

[in] Used for ANSI,ISO-0 and ISO-1 algorithm to build the

formatted PIN. For ANSI and ISO-0 the PAN (Primary Account Number)is used .for ISO-1 a ten digital transaction field is required. If not used a UNLL is required.

lpXORData
[in] Reserved; must be NULL.
If the formatted PIN is encrypted twice to build the resulting PIN block,this data can be used to modify the result of the first encryption by an XOR-operation.

byPadding
[in] Specifies the padding character .

byPinLen
[in] the byte number of pin.

lpPinBlock
[out] PinBlock data.

lpPinBlockLen
[out] the byte number of the PinBlock

Return Values

If the function succeeds, the return value is zero.

If the function fails, the return value is nonzero. To get extended error information, see *Error Code List*.

3.9. ZT_PIN_Crypt

Description

This command can be used for one of the below status:

Encrypt or decrypt by using specified encryption mode;

Calculate the MAC;

Generate random data;

Prototype

```
int   WINAPI ZT_PIN_Crypt(WORD     wKeyId,
                          LPBYTE   lpEnKeyValue,
                          BYTE     byEnKeyValueLen,
                          WORD     wVI,
                          WORD     wMode,
                          WORD     wAlgorithm,
                          WORD     wStartValueKeyId,
                          LPBYTE   lpStartValue,
                          LPBYTE   lpCryptData，
                          WORD     wCryptDataLen,
                          LPBYTE   lpOutCryptData,
                          LPWORD   lpOutCryptDataLen)
```

Parameters

wKeyId

[in] Specifies the index of the stored key, The value range is 1 to 599. if the wMode equals WFS_PIN_MODERANDOM,this value ignored.

lpEnKeyValue

[in] Reserved; must be zero.

byEnKeyValueLen

[in] Reserved; must be zero.

wVI

[in] Reserved; must be zero.

wMode

[in] When encrypt or decrypt data,the value as below:

Value	Crypt mode
1	WFS_PIN_MODEENCRYPT
2	WFS_PIN_MODEDECRYPT
3	WFS_PIN_MODERANDOM

When apply to MACing,the value as below:

Value	Crypt mode
0x41	- ISO 8731-2 or ANSI X 9.9 MAC Algorithm
0x42	- ISO 9797-1 MAC Algorithm 3, Padding Method 1
0x43	- ISO 16609 MAC Algorithm 1, Padding Method 1
0x44	- MAC for PBOC or ANSI X 9.9, Padding Method 3
0x45	- MAC for China Union Pay

wAlgorithm

[in] Specifies the encryption algorithm. Descriptions are as follows:

Value	Algorithm
0x0001	WFS_PIN_CRYPTDESECB
0x0002	WFS_PIN_CRYPTDESCBC
0x0004	WFS_PIN_CRYPTDESCFB (not support)
0x0008	WFS_PIN_CRYPTRSA
0x0010	WFS_PIN_CRYPTECMA (not support)
0x0020	WFS_PIN_CRYPTDESMAC
0x0040	WFS_PIN_CRYPTTRIDESECB
0x0080	WFS_PIN_CRYPTTRIDESCBC
0x0100	WFS_PIN_CRYPTTRIDESCFB (not support)
0x0200	WFS_PIN_CRYPTTRIDESMAC
0x0400	WFS_PIN_CRYPTMAAMAC (not support)

wStartValueKeyId

[in] Reserved; must be zero.

specifies the index of the stored key used to decrypt the lpStartVale to obtain the initialization vector. If it is null, lpStartVale is used as the initialization vector. this value is ignored if wMode equals to WFS_PIN_MODERANDOM.

lpStartValue

[in] DES and Triple DES initialization vetor for CBC/CFB encrypting and MACing .if this parameter is null wStartValueKeyId is used as the start value. If wStartValueKeyId is also null,the default value for CBC/CFB/MAC is 16 hex digitals 0x0. this value is ignored if wMode equals to WFS_PIN_MODERANDOM.

lpCryptData

[in] Pointer to the data to be encrypted ,decrypted ,or MACed; when the wMode equals WFS_PIN_MODERANDOM,this value ignored.

The byte length of lpCryptData must be 8*N(N=1,2….),not enough padded by specifies the padding character.

wCryptDataLen

[in] the byte length of lpCryptData;must be 8*N(N=1,2….)

lpOutCryptData

[out] Pointer to the data to be encrypted ,decrypted ,or MACed

lpOutCryptDataLen

[out] the byte length of the cryptdata;

Return Values

If the function succeeds, the return value is zero.

If the function fails, the return value is nonzero. To get extended error information, see *Error Code List*.

3.10. ZT_PIN_InitialDevice

Description

Initialize the Pinpad device . All keys will be erased.

This command can be used only one time after the PinPad device is installed to ATM.

Prototype

```
int WINAPI ZT_PIN_InitialDevice(BYTE    byMode)
```

Parameters

byMode
[in] Reserved. The value is zero.

Return Values

If the function succeeds, the return value is zero.

If the function fails, the return value is nonzero. To get extended error information, see *Error Code List*.

3.11. ZT_PIN_ReadKeyAttribute

Description

Read key attributes and key check value of specified key.

Prototype

```
int WINAPI ZT_PIN_ReadKeyAttribute(
                    WORD        wKeyId,
                    LPWORD      lpKeyAttributes,
                    LPBYTE        lpKeyCheckValue)
```

Parameters

wKeyId
[in] Specifies the index of the stored key, The value range is 1 to 599.

lpKeyAttributes
[out]Key attributes. Fiexd length 1 word.

lpKeyCheckValue
[out] Key check value.

Return Values

If the function succeeds, the return value is zero.

If the function fails, the return value is nonzero. To get extended error information, see *Error Code List*.

3.12. ZT_PIN_ReadKCV

Description

Read key check value of specified key.

Prototype

```
int WINAPI ZT_PIN_ReadKCV(WORD wKeyId, LPBYTE lpKeyCheckValue)
```

Parameters

wKeyId
[in] Specifies the index of the stored key, The value range is 1 to 599.

lpKeyCheckValue

[out] Key check value.

Return Values

If the function succeeds, the return value is zero.

If the function fails, the return value is nonzero. To get extended error information, see *Error Code List*.

3.13. ZT_PIN_WriteUserData

Description

Write 256 bytes block into specified block address of pinpad's user data area.

Prototype

```
int WINAPI    ZT_PIN_WriteUserData(BYTE    byBlockAddress,
                                   LPBYTE lpUserData)
```

Parameters

byBlockAddress
[in] Block address; range from 0 to 99

lpUserData
[in] data to be written into user data area;must be 256 bytes.

Return Values

If the function succeeds, the return value is zero.

If the function fails, the return value is nonzero. To get extended error information, see *Error Code List*.

3.14. ZT_PIN_ReadUserData

Description

Read 256 bytes block from the specified block address of pinpad's user data area.

Prototype

```
int WINAPI    ZT_PIN_ReadUserData(BYTE    byBlockAddress,
                                  LPBYTE lpUserData)
```

Parameters

byBlockAddress
[in] Block address; range from 0 to 99

lpUserData
[in] data to be read from user data area;must be 256 bytes.

Return Values

If the function succeeds, the return value is zero.

If the function fails, the return value is nonzero. To get extended error information, see *Error Code List*.

3.15. ZT_PIN_WriteKeyValue

Description

Modify all key values of the pinpad.

Prototype

```
int WINAPI    ZT_PIN_WriteKeyValue(BYTE      bMode,
                                   LPBYTE lpKeyValueList)
```

Parameters

bMode
[in] Set keyboard code Mode, the value as below:
0x31 Set keyboard code for input ~~PIN~~ TEXT
0x32 Set keyboard code for input ~~KEY~~ PIN
0x33 Set keyboard code for input ~~TEXT~~ KEY.

lpKeyValueList
[in] 24 bytes key value.The key position layout is on the EPP as follows:

00	**01**	**02**	**03**
04	**05**	**06**	**07**
08	**09**	**0A**	**0B**
0C	**0D**	**0E**	**0F**

Extended function key position layout

10	**11**	**12**	**13**
14	**15**	**16**	**17**

Key value range is 0x00~0xFF，except 2 values for the other functions：

0x00——blank key value ,no response in any entry mode

0x80——Overtime flag if press one key for a long time

Return Values

If the function succeeds, the return value is zero.

If the function fails, the return value is nonzero. To get extended error information, see *Error Code List*.

3.16. ZT_PIN_ReadEPPInfo

Description

Read pinpad information.

Prototype

```
int WINAPI ZT_PIN_ReadEPPInfo(LPBYTE szEPPInfo
                              LPBYTE lpInfoLen)
```

Parameters

szEPPInfo

[out] firmware version; 16 bytes

lpInfoLen
[out] specifies the byte number of the *szEPPInfo*

Return Values

If the function succeeds, the return value is zero.

If the function fails, the return value is nonzero. To get extended error information, see *Error Code List*.

3.17. ZT_PIN_Backspace

Description

Backspace while PIN or key input .

Prototype

```
int WINAPI ZT_PIN_Backspace()
```

Parameters

None.

Return Values

If the function succeeds, the return value is zero.

If the function fails, the return value is nonzero. To get extended error information, see *Error Code List*.

3.18. ZT_PIN_GetDeviceStatus

Description

Get pinpad device status.

Prototype

```
int WINAPI ZT_PIN_GetDeviceStatus(LPBYTE lpStatus)
```

Parameters

lpStatus

[out]Device status;1 byte. Details as below:

0: online

1: offline

2: battery lack of power

Return Values

If the function succeeds, the return value is zero.

If the function fails, the return value is nonzero. To get extended error information, see *Error Code List*.

3.19. ZT_PIN_UpdatePassword

Description

Update administrator password.

Prototype

```
int WINAPI ZT_PIN_UpdatePassword(
                    WORD    wKeyId,
                    BYTE    byMode)
```

Parameters

wKeyId
[int] the value as below:
901 PWD1 for manual input DES-KEY
902 PWD2 for manual input DES-KEY
903 PWD3 for manual input DES-KEY
904 Password1 for install remove auth EPP
905 Password 2 for install remove auth EPP

byMode
[int] Update Mode, the value as below:
0x30 Manual input
0x31 Command input.

Return Values

If the function succeeds, the return value is zero.

If the function fails, the return value is nonzero. To get extended error information, see *Error Code List*.

3.20. ZT_PIN_RemoveInstall

Description

Authorized to remove or install EPP.

Prototype

```
int WINAPI ZT_PIN_RemoveInstall(BYTE byMode)
```

Parameters

byMode
[in]

0: It not need installed EPP to ATM, EPP can do every function after Imperative Install EPP.

1：EPP can do every function except used DES-key after it remove from ATM. All DES-key is locked, but manual input key from EPP for loading key.

2：EPP must re-installed to ATM, EPP can do every function

after it authorize remove from ATM

Return Values

If the function succeeds, the return value is zero.

If the function fails, the return value is nonzero. To get extended error information, see *Error Code List*.

3.21. ZT_PIN_VirtualKeyInput

Description

Pass string to EPP use for PIN.

Prototype

```
int WINAPI ZT_PIN_VirtualKeyInput(char *KeyTable, int KeyTableLen)
```

Parameters

KeyTable
[in]the string send to EPP, consist of '0-F', these num map of EPP position

0	**1**	**2**	**3**
4	**5**	**6**	**7**
8	**9**	**A**	**B**
C	**D**	**E**	**F**

KeyTableLen
[in] the string length

Return Values

If the function succeeds, the return value is zero.

If the function fails, the return value is nonzero. To get extended error information, see *Error Code List*.

3.22. ZT_PIN_ImportRsaKey

Description

Download the RSA Key to EPP.

Prototype

```
int WINAPI ZT_PIN_ImportRsaKey (WORD wKeyID,
                                WORD wKeyAttr,
                                WORD wSignKeyID,
                                BYTE bSignAlgorithm,
                                LPBYTE lpKeyTag,
                                WORD wKeyTagLen,
                                BYTE bKeyCheckMode,
                                LPBYTE lpKCV,
                                LPWORD wKCVLen)
```

Parameters

wKeyID

[in] Specifies the ID of key being loaded

wKeyAttr

[in] RSA Key's attribute, the Vendor key is 0x8050, other's is 0x8060

wSignKeyID

[in] Specifies the Key ID of verify the signature key. Only verify signature Used Vendor Public Key

bSignAlgorithm

[in] Defines algorithm/method used to generate the public key check value/thumb print, The check value can be used to verify that the public key has been imported correctly.

0x30 = Not Signature

0x31 = RSASSA_PKCS1_V1_5

lpKeyTag

[in] Defines the algorithm used to generate the signature specified in Signature Key.

wKeyTagLen

[in] Key tag in DER encoding.

bKeyCheckMode

[Out] 0x00 Not Key Check Value

0x01 RSA_KCV_SHA1

lpKCV

[out] Contains the public key check value as defined.

wKCVLen

[out] Contains the public key check value as defined length.

Return Values

If the function succeeds, the return value is zero.

If the function fails, the return value is nonzero. To get extended error information, see Error Code List.

3.23. ZT_PIN_ExportRsaKey

Description

Export the RSA Public Key to EPP.

Prototype

```
int WINAPI ZT_PIN_ExportRSAKey(WORD wKeyId,
                               WORD wsigKeyId,
                               LPBYTE lpSigAlgorithm,
                               LPBYTE lpPublicKey,
                               LPWORD wPublicKeyLen)
```

Parameters

wKeyId

[in] Key ID symmetric key will be stored.

wSigKeyId,

[in] Signature SKvnd of Vendor or signature SKsepp of EPP.

lpSigAlgorithm,

[out]Specifies Signature Algorithm.

0x30 Not Signature

0x31 RSASSA_PKCS1_V1_5

lpPublicKey,

[out]Specifies the Public Key tag in DER encoding.

wPublicKeyLen,

[out]specifies the byte number of lpPublicKey

Return Values

If the function succeeds, the return value is zero.

If the function fails, the return value is nonzero. To get extended error information, see Error Code List.

3.24. ZT_PIN_ImportDesKey

Description

Load a key that is either a single or double DES length key into EPP.

Prototype

```
int WINAPI ZT_PIN_ImportDesKey(WORD wKeyID,
                               WORD wKeyAttr,
                               WORD wDecryptSK,
                               BYTE bSignAlgorithm,
                               BYTE bVerifyMode,
                               BYTE bKCM,
                               LPBYTE lpKCV,
                               WORD wHOSTPK,
                               BYTE bEncipherAlg,
                               BYTE bRandMode,
                               LPBYTE lpEncryptedKeytag,
```

WORD wKeyTagLen,

LPBYTE lpEppKCV,

LPWORD wEppKCVLen)

Parameters

wKeyID

[in] Key ID stored location

wKeyAttr

[in] DES Key attributes

wDecryptSK,

[in] Only EPP SK decrypted symmetric key

bSignAlgorithm,

[in] Specifies Signature Algorithm

0x30 Not Signature

0x31 RSASSA_PKCS1_V1_5

bVerifyMode,

[in] Specifies command with KCV

0x30 Do not check KCV

0x31 Check KCV

bKCM,

[in] calculate KCV Mode

0x30 The key to encrypt with the first 8 bytes of itself

0x31 The key to encrypt with 8 byte 0x00

lpKCV,

[in] Data calculated KCV, if with KCV in command

wHOSTPK,

[in] Option HOST PK verify signature

bEncipherAlg,

[in] 0x31 RSAES_PKCS1_V1_5

bRandMode,

[in] 0x31 Need add 8 bytes random Data to digest for authention

lpEncryptedKeytag,

[in] Encrypted DES-Key tag in DER encoding

wKeyTagLen

[in] Encrypted DES-Key tag in DER encoding's length

lpEppKCV,

[in] Key Check Value

wEppKCVLen

[in] Key Check Value 's length

Return Values

If the function succeeds, the return value is zero.

If the function fails, the return value is nonzero. To get extended error information, see Error Code List.

3.25. ZT_PIN_ ExportDesKey

Description

This command instructs EPP to generate a random DES key of given length and to export it enciphered with a previously loaded RSA public key using the specified encryption algorithm.

Prototype

```
int WINAPI ZT_PIN_ExportDesKey(WORD wKeyId,
                               WORD wKeyAttr,
                               WORD wSignKey,
                               BYTE bSignAlgorithm,
                               WORD wPKEncrypt,
                               BYTE bGenMode,
                               BYTE bHOSTCKM,
                               LPBYTE lpEPPRand,
                               LPBYTE lpHOSTRand,
                               LPBYTE lpHostID,
                               WORD wHOSTSignID,
                           LPBYTE lpEPPSNpackage,
                         WORD wEPPSNpackageLen,
                        LPBYTE lpEPPSymmetricKey,
                      LPWORD wEPPSymmetricKeyLen)
```

Parameters

wKeyId

[in] Key ID symmetric key will be stored

wKeyAttr

[in] DES Key attributes

wSignKey

[in] Only signature SKsepp of EPP Public Key

bSignAlgorithm

[in] Specifies Signature Algorithm

0x30 Not Signature

0x31 RSASSA_PKCS1_V1_5

wPKEncrypt

[in] Option HOST PK to encrypt Symmetric Key, KeyID range 606~619

bGenMode

[in]

0x30 Not need check EPP ID

0x31 need check EPP ID

bHOSTCKM

[in]

0x30 Not use HOST ID

0x31 Use Host ID

lpEPPRand

[in] A random number RHOST generated by EPP

lpHOSTRand

[in] A random number RHOST generated by HOST

lpHostID

[in] Host ID

wHOSTSignID

[in] Host signature host public key ID

lpEPPSNpackage

[in] Sign EPP SN with Host signature private key . Signature is calculated:

EPP SN Tag

EPP SN

wEPPSNpackageLen

[in] Sign EPP SN signature Len

lpEPPSymmetricKey

[out] EPP Symmetric key tag in DER encoding

wEPPSymmetricKeyLen

[out] EPP Symmetric key tag in DER encoding's length

Return Values

If the function succeeds, the return value is zero.

If the function fails, the return value is nonzero. To get extended error information, see Error Code List.

3.26. ZT_PIN_RawRsaEncrypt

Description

Encrypt specified data using supplied RSA key.

Command performs following actions:

- RSA Encrypt specified data using supplied RSA key
- Returns data

Prototype

int WINAPI ZT_PIN_RawRsaEncrypt(

WORD wRSAModulusLen,

LPBYTE lpRSAModulus,

```
                                WORD wRSAExponentLen,
                                LPBYTE lpRSAExponent,
                                WORD wInDataLen,
                                LPBYTE lpInData,
                                LPWORD lpEncrptedDataLen,
                                LPBYTE lpEncrptedData)
```

Parameters

wRSAModulusLen

[in] RSA public key modulus length

lpRSAModulus

[in] RSA public key modulus

wRSAExponentLen

[in] RSA public key exponent length

lpRSAExponent

[in] RSA public key exponent

wInDataLen

[in] User data length

lpInData

[in] Encrypt Data

lpEncrptedDataLen

[out] Encrypted data's length

lpEncrptedData

[out] Encrypted data

Return Values

If the function succeeds, the return value is zero.

If the function fails, the return value is nonzero. To get extended error information, see Error Code List.

3.27. ZT_PIN_RSAEncrypt

Description

Encrypt block of data using specified RSA key.

Prototype

```
int WINAPI ZT_PIN_RSAEncrypt(WORD wKeyId,
                             LPBYTE lpInputData,
                             WORD wInputDataLen,
                             LPBYTE lpOutData,
```

LPWORD OutDataLen)

Parameters

wKeyId

[in] RSA Key ID

lpInputData

[in] Encrypt Data

wInputDataLen

[in] User data length

lpOutData

[out] Encrypted data

OutDataLen

[out] Encrypted data length

Return Values

If the function succeeds, the return value is zero.

If the function fails, the return value is nonzero. To get extended error information, see Error Code List.

3.28. ZT_PIN_CreateOffset

Description

Create PIN Offset with specified Validation Data.

Prototype

```
int WINAPI ZT_PIN_CreateOffset(WORD wKeyId,
                               WORD wEnKeyId,
                               WORD wVI,
                               LPSTR lpValidationData,
                               BYTE byValDigits,
                               LPSTR lpDecTable,
                               LPSTR lpPINOffset);
```

Parameters

wKeyId

[in] Id of the key which have the 0xA000 attribute,also called PIN Key.

wEnKeyId

[in] Not used

wVI

[in] Variant Index.

lpValidationData

[in] Validation data. Must be hexadecimal character, less than 16 characters.

byValDigits

[in] Number of Validation Data Digits to be used for PIN Offset creation

lpDecTable

[in] ASCII decimalization table(16 character string containing characters ‘0’ to ‘9’).Used to convert the hexadecimal digits(0x0 to 0xF) of the encrypted validation data to decimal digits(0x0 to 0x9)

lpPINOffset

[out] The Computed PIN Offset returned

Return Values

If the function succeeds, the return value is zero.

If the function fails, the return value is nonzero. To get extended

error information, see Error Code List.

3.29. ZT_PIN_LocalDes

Description

Validate PIN used specified Validation data and PIN offset.

Prototype

```
int WINAPI ZT_PIN_LocalDes(    WORD wKeyId,
                               WORD wEnKeyId,
                               WORD wVI,
                               LPSTR lpValidationData,
                               BYTE byValDigits,
                               LPSTR lpDecTable,
                               LPSTR lpPINOffset);
```

Parameters

wKeyId

[in] Id of the key which have the 0xA000 attribute,also called PIN Key.

wEnKeyId

[in] Not used

wVI

[in] Variant Index.

lpValidationData

[in] Validation data. Must be hexadecimal character,less than 16 characters.

byValDigits

[in] Number of Validation Data Digits to be used for PIN Offset Validation.

lpDecTable

[in] ASCII decimalization table(16 character string containing characters ‘0’ to ‘9’).Used to convert the hexadecimal digits(0x0 to 0xF) of the encrypted validation data to decimal digits(0x0 to 0x9)

lpPINOffset

[out] The Computed PIN Offset for validation.

Return Values

If the function succeeds, the return value is zero.

If the function fails, the return value is nonzero. To get extended error information, see Error Code List.

3.30. ZT_PIN_StoreKey

Description

Store the key in the encryption module. The key can be inputted from the EPP with the command ZT_PIN_SetEntryMode(0x32), accompanying "key encryption key".

Prototype

```
int WINAPI ZT_PIN_StoreKey(WORD wKeyId,
                           WORD wUse,
                           BYTE byMode,
                           LPBYTE lpKCV,
                           LPBYTE lpKCVL)
```

Parameters

wKeyId

[in] Specifies the index of key being loaded.The value range is 1 to 599.

wUse

[in] Specifies the type of access for which the can be used as a combination of the following flags:

Value	Meaning
WFS_PIN_USEKEYENCKEY (0x0020)	Key is used as key encryption key.
WFS_PIN_USEFUNCTION (0x0002)	Key can be used for PIN function.
WFS_PIN_USEMACING (0x0004)	Key can be used for MACing.
WFS_PIN_USECRYPT (0x0001)	Key can be used for encryption/decryption

In general, the type of access must be unique. Only WFS_PIN_USEMACING and WFS_PIN_USECRYPT can be as a combination

byMode
[in] ImportKey mode;the value as below:
0x30:direct stroed
0x31:XOR with existing key

lpKCV
[out] Fixed length 8 bytes; Contains the key verification code data that can be used for verification of the loaded key.

lpKCVL
[out] Specifies the byte number of the *lpKCV*.

Return Values

If the function succeeds, the return value is zero.

If the function fails, the return value is nonzero. To get extended error information, see *Error Code List*.

Remarks

The value of Every key must be different from Others.

4. Annexes

4.1. Error Code List

Value	Error code	Meaning
STATUS_SUCCESS	0x00	No error – status successful
STATUS_INVALID_COMMAND_LENGTH	0x01	Command length not comply with definition
STATUS_INVALID_KEYID	0x02	KeyID(s) is outside of valid range
STATUS_INVALID_WRITE_MODE	0x03	Invalid Write Mode Specified
STATUS_KEYID_NOT_EXIST	0x04	Key with specified ID doesn't exist
STATUS_KEYID_LOCKED	0x05	Key with specified ID is locked
STATUS_INVALID_KEY_LENGTH	0x06	The Key to be stored has invalid length (not 8 ,16 or 24 bytes or less than 16 bytes with MK Attributes or greater than 8 bytes with IV Attributes)
STATUS_INVALID_KEY_ATTRIBUTES	0x07	Key ID has wrong attribute
STATUS_INVALID_IV_ATTRIBUTES	0x08	IV has wrong attribute

STATUS_KEY_ATTRIBUTES_NOT_MATCHED	0x09	Key Attribute not Match the exist Attribute
STATUS_KCV_VERIFICATION_FAIL	0x0A	Key Check Value Verification Failure
STATUS_INVALID_KEY_VALUE	0x0B	Other KeyId Has the Same Key Value
STATUS_KEY_USEVIOLATION	0x10	Key with specified ID Useviolation
STATUS_KEY_NOTALLOWED	0x11	Key with specified ID Not Allowed to do that
STATUS_INVALID_CHECK_MODE	0x12	Check Mode Invalid
STATUS_INVALID_VERIFY_MODE	0x13	Verify Mode Invalid
STATUS_INVALID_MACING	0x14	KCV verification failed
STATUS_INVALID_AUTHENTICATION_MODE	0x15	Macing Invalid Or Verify Failure
STATUS_INVALID_MODE	0x16	Invalid Mode or Entry Mode specified
STATUS_INVALID_DATA_SPECIFIED	0x17	Invalid data - values out of range
STATUS_INVALID_LENGTH_	0x18	Invalid Length or Offset+Length -

OR_SUM		out of range
STATUS_INVALID_OFFSET_LENGTH	0x19	Invalid Offset or Length - out of range
STATUS_INVALID_PIN_LENGTH	0x20	Invalid PIN Length
STATUS_PIN_VERIFICATION_FAIL	0x21	PIN verification failed
STATUS_PIN_ENCRYPTION_SUSPENDED	0x22	PIN Encryption Suspended ,try Pin after one hour
STATUS_PIN_NOPIN	0x23	No Pin
STATUS_INVALID_PIN2_LENGTH	0x24	The password in key buffer has wrong length
STATUS_PIN2_VERIFICATION_FAIL	0x25	The password in key buffer verification failed
STATUS_INVALID_MODULUS_LENGTH	0x31	Invalid modulus length specified
STATUS_INVALID_EXPONENT_LENGTH	0x32	Invalid exponent length specified
STATUS_INVALID_PKCS_STRUCTURE	0x33	Invalid structure received
STATUS_INVALID_PKCS_PADDING	0x34	Invalid PKCS Padding

STATUS_INVALID_SIGNATURE_LENGTH	0x35	Invalid Signature Length
STATUS_SIGNATURE_VERIFICATION_FAIL	0x36	Signature verification failed
STATUS_KEY_NOT_LOADED	0x37	Key with specified ID not Loaded
STATUS_INVALID_CERTIFICATE_FROMAT	0x41	INVALID CERTIFICATE FROMAT
STATUS_INVALID_CERTIFICATE_VERSION	0x42	INVALID CERTIFICATE VERSION
STATUS_INVALID_CERTIFICATE_VALIDITY	0x43	INVALID CERTIFICATE ISSUER
STATUS_INVALID_CERTIFICATE_ISSUER	0x44	INVALID CERTIFICATE VALIDITY
STATUS_INVALID_CERTIFICATE_SUBJECT	0x45	INVALID CERTIFICATE SUBJECT
STATUS_INVALID_CERTIFICATE_ALGOR	0x46	INVALID CERTIFICATE ALGOR
STATUS_CERTIFICATE_NOT_EXIST	0x47	Certificate with specified ID Not Exist
STATUS_INVALID_DATA_DISABLEED	0x50	Invalid Date Disableed
STATUS_INVALID_USERBLO	0x51	Invalid User Block Address

CK_ADDRESS		specified
STATUS_INVALID_SERIALNO_SPECIFIED	0x52	Invalid Serial Number specified
STATUS_ALGORITHMNOTSUPP	0x71	ALGORITHM UNSUPP
STATUS_COMMANDUPSUPP	0x72	COMMAND UNSUPP
STATUS_ERROR_INTER_ERROR	0x73	HARDWARE ERROR OR INTER ERROR
STATUS_COMMAND_LOCK	0x74	Command Lock :Not allow Execte
STATUS_KEYCOMMAN_MAC	0x75	KEY_COMMAND_MAC value verification error
STATUS_AUTHENTICATION_FAILED	0x80	Authentication failure
STATUS_NOT_AUTHENTE	0x81	Key with specified ID doesn't authentication
STATUS_EPP_NOT_INITIALIZED	0x82	EPP Keypair and/or Serial# signature not loaded
STATUS_EPP_ALREADY_INITIALIZED	0x83	EPP Keypair and/or Serial# signature already loaded
STATUS_CERTIFICATE_ALREADY	0x84	EPP Certification already loaded

STATUS_GETPIN_TIMEOUT	0x85	Wait for a pressed key over time in PIN ENTRY mode or KEY ENTRY mode.
API_PARAMETER_ERROR	0x90	Parameter error
API_PINFORMAT_UNSUPP	0x91	Invalid pin format
API_ALGORITHM_UNSUPP	0x92	Invalid crypt algorithm
API_EXCUTE_FAIL	0x93	Other execute error
ERR_COMM_FAIL	-1	Communication failure/ Open com failed
ERR_COMM_TIMEOUT	-2	Operation time out
ERR_COMM_CANCELED	-3	Operation cancel
ERR_COMM_PARAM	-4	Parameter Error
ERR_COMM_NODEFINED	-100	Other communication error