FEITIAN

Upgrade - Additional Functions in APIs for ROCKEY6SMART

V2.0

Feitian Technologies Co., Ltd.

Website: www.FTsafe.com

Revision History:

Date	Revision	Description
Sep. 2008	V1.0	Release of the first version
Dec. 2009	V2.0	Release of the second version

Software Developer's Agreement

All Products of Feitian Technologies Co., Ltd. (Feitian) including, but not limited to, evaluation copies, diskettes, CD-ROMs, hardware and documentation, and all future orders, are subject to the terms of this Agreement. If you do not agree with the terms herein, please return the evaluation package to us, postage and insurance prepaid, within seven days of their receipt, and we will reimburse you the cost of the Product, less freight and reasonable handling charges.

1. Allowable Use – You may merge and link the Software with other programs for the sole purpose of protecting those programs in accordance with the usage described in the Developer's Guide. You may make archival copies of the Software.

2. Prohibited Use – The Software or hardware or any other part of the Product may not be copied, reengineered, disassembled, decompiled, revised, enhanced or otherwise modified, except as specifically allowed in item 1. You may not reverse engineer the Software or any part of the product or attempt to discover the Software's source code. You may not use the magnetic or optical media included with the Product for the purposes of transferring or storing data that was not either an original part of the Product, or a Feitian provided enhancement or upgrade to the Product.

3. Warranty – Feitian warrants that the hardware and Software storage media are substantially free from significant defects of workmanship or materials for a time period of twelve (12) months from the date of delivery of the Product to you.

4. Breach of Warranty – In the event of breach of this warranty, Feitian's sole obligation is to replace or repair, at the discretion of Feitian, any Product free of charge. Any replaced Product becomes the property of Feitian.

Warranty claims must be made in writing to Feitian during the warranty period and within fourteen (14) days after the observation of the defect. All warranty claims must be accompanied by evidence of the defect that is deemed satisfactory by Feitian. Any Products that you return to Feitian, or a Feitian authorized distributor, must be sent with freight and insurance prepaid.

EXCEPT AS STATED ABOVE, THERE IS NO OTHER WARRANTY OR REPRESENTATION OF THE PRODUCT, EXPRESSED OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE.

5. Limitation of Feitian's Liability – Feitian's entire liability to you or any other party for any cause whatsoever, whether in contract or in tort, including negligence, shall not exceed the price you paid for the unit of the Product that caused the damages or are the subject of, or indirectly related to the cause of action. In no event shall Feitian be liable for any damages caused by your failure to meet your obligations, nor for any loss of data, profit or savings, or any other consequential and incidental damages, even if Feitian has been advised of the possibility of damages, or for any claim by you based on any third-party claim.

6. Termination – This Agreement shall terminate if you fail to comply with the terms herein. Items 2, 3, 4 and 5 shall survive any termination of this Agreement.

PREFACE

This document gives detailed description on the additional functions provided in APIs for ROCKEY6SMART during the update process.

Contents

Chapter 1. Additional Functions in APIs for ROCKEY6SMART

1.1 Get Volume and Manufacturer Info

```
EXTERN_C int WINAPI DIC_GetCardInfo(int hic, char strVolume[17], char strManufactureInfo[16]) ;
```

Function description:

Get volume and manufacturer information

Parameters:

strVolume:	[out]return volume of the dongle
strManufactureInfo:	[out]return the manufacturer information inside the dongle

These two parameters can only be changed by the manufacturer, not the client.

Return value:

Please refer to Chapter 2 Error Code List.

1.2 Get Producing Time, Serial No, Sales Time and COS version

```
EXTERN_C int WINAPI DIC_GetHardwareInfo(int hic,
                                        DWORD *pdwProducedTime,
                                        DWORD *pdwSerial,
                                        DWORD *pdwSalesTime,
                                        DWORD *pdwCOSVersion);
```

Function description:

Get producing time, serial no, sales time and COS version of the dongle.

Parameters:

pdwProduceTime:	[out]return producing time
pdwSerial:	[out]return serial no
pdwSalesTime:	[out]return sales time

pdwCOSVersion:	[out]return COS version

These four parameters are set by the manufacturer and cannot be changed.

Return value:

Please refer to Chapter 2 Error Code List.

1.3 Get Zone Code, Agent Code and Two User Codes

```
EXTERN_C int WINAPI DIC_GetManagerCode(int hic,
                                       WORD *pwZoneCode,
                                       WORD *pwAgentCode,
                                       WORD *pwUser1Code,
                                       WORD *pwUser2Code);
```

Function description:

Get zone code, agent code and two user codes of the dongle.

Parameters:

pwZoneCode:	[out]return zone code
pwAgentCode:	[out]return agent code
pwUser1Code:	[out] return user 1 code
pwUser2Code:	[out] return user 2 code

Return value:

Please refer to Chapter 2 Error Code List.

1.4 Get Password Verification Status

```
EXTERN_C int WINAPI DIC_GetPasswordVerifyState(int hic, int *piVerifyState);
```

Function description:

Get password verification status, return value piVerifyState.

Parameters:

piVerifyState:	[out]password verification status

Flag	Meaning
NO_PASSED	Password verification not passed
SUPER_PASSWORD_PASSED	Super password verification passed
REMOTE_PASSWORD_PASSED	Remote update password verification passed
SUPER_REMOTE_PASSWORD_PASSED	Super and Remote update password verification passed

Please refer the definition of the constants for these values.

Return value:

Please refer to Chapter 2 Error Code List.

1.5 Get Remote Update Identity and Password

```
EXTERN_C int WINAPI DIC_GetRemoteInfo(int hic, DWORD *pdwFlag, BYTE abyPassword[8]);
```

Function description:

Get remote update identity and remote update password.

Parameters:

pdwFlag:	[out]return remote update identity
abyPassword:	[out]return remote update password

Return value:

Please refer to Chapter 2 Error Code List.

1.6 Set Remote Update Identity and Password

```
EXTERN_C int WINAPI DIC_SetRemoteInfo(int hic, DWORD dwFlag, const BYTE abyPassword[8]);
```

Function description:

Set remote update identity and remote update password.

Parameters:

dwFlag:	[in]set remote update identity
abyPassword:	[in]set remote update password

Return value:

Please refer to Chapter 2 Error Code List.

1.7 Verify Remote Update Identity and Password

```
EXTERN_C int WINAPI DIC_CheckRemoteInfo(int hic, DWORD dwFlag, const BYTE byPassword[8]);
```

Function description:

Verify remote update identity and remote update password.

Parameters:

dwFlag:	[in]the remote update identity that needs to be verified
abyPassword:	[in]the remote update password that needs to be verified

Return value:

Please refer to Chapter 2 Error Code List.

1.8 Get ID, Type, Attribute, Security Level and Name of the Current Directory

```
EXTERN_C int WINAPI DIC_GetCurrentDir(int hic,
                                      WORD *pwDirID,
                                      BYTE *pbyDirType,
                                      BYTE *pbyDirAttribute,
                                      BYTE *pbyDirSecurity,
                                      char strDirName[17]);
```

Function description:

Get ID, Type, attribute, security level and name of the current directory.

Parameters:

pwDirID:	[out]return ID of the current directory
pbyDirType:	[out] return type of the current directory
pbyDirAttribute:	[out] return attributes of the current directory
pbyDirSecurity:	[out] return security level of the current directory
strDirName:	[out] return name of the current directory

Return value:

Please refer to Chapter 2 Error Code List.

1.9 Set Current Directory

```
EXTERN_C int WINAPI DIC_SetCurrentDir(int hic, WORD wDirID, char strDirName[]);
```

Function description:

Set current directory as the directory of wDirID.

Parameters:

wDirID:	[in]directory ID
strDirName:	[in]directory name

Note: this function can be used to set the current directory by a name or an ID. Normally only one parameter needs to be specified, the other can be set as NULL or 0. If both parameters are used, they should aim at one directory.

Return value:

Please refer to Chapter 2 Error Code List.

1.10 Get ID, Type, Attributes, Security Level and Name of the Current File

```
EXTERN_C int WINAPI DIC_GetCurrentFile(int hic,
                                       WORD *pwFileID,
                                       BYTE *pbyFileType,
                                       BYTE *pbyFileAttribute,
                                       BYTE *pbyFileSecurity,
                                       char strFileName[18]);
```

Function descriptions:

Get ID, type, attributes, security level and name of the current file.

Parameters:

pwFileID:	[out]return ID of the current file
pbyFileType:	[out] return type of the current file
pbyFileAttribute:	[out] return attributes of the current file

pbyFileSecurity:	[out] return security level of the current file
strFileName:	[out] return name of the current file

Return value:

Please refer to Chapter 2 Error Code List.

1.11 Set Current File

```
EXTERN_C int WINAPI DIC_SetCurrentFile(int hic, WORD wFileID, char strFileName[]);
```

Function description:

Set the current file as the file identified by wFileID

Parameters:

wFileID:	[in]file ID
strFileName:	[in]file name

Note: this function can be used to set the current file by using an ID or a file name. Normally one parameter is used; the other parameter can be set as NULL or 0. If both parameters are used, they should aim at one file.

Return value:

Please refer to Chapter 2 Error Code List.

1.12 Get ID, Type, Attribute, Security Level and Name of the Parent Directory

```
EXTERN_C int WINAPI DIC_GetParentDir(int hic,
                                     WORD *pwDirID,
                                     BYTE *pbyDirType,
                                     BYTE *pbyDirAttribute,
                                     BYTE *pbyDirSecurity,
                                     char strDirName[17]);
```

Function description:

Get ID, type, attribute, security level and name of the parent directory

Parameters:

pwDirID:	[out]return ID of the parent directory
pbyDirType:	[out] return type of the parent directory
pbyDirAttribute:	[out] return attribute of the parent directory
pbyDirSecurity:	[out] return security level of the parent directory
strDirName:	[out] return name of the parent directory

Return value:

Please refer to Chapter 2 Error Code List.

1.13 Return to the Parent Directory

```
EXTERN_C int WINAPI DIC_SetParentDir(int hic);
```

Function description:

Return to the parent directory

Parameters: None

Return value:

Please refer to Chapter 2 Error Code List.

1.14 List First Directory or File under the Current Directory

```
EXTERN_C int WINAPI DIC_FindFirstFileOrDir(int hic,
                                           int *piPosition,
                                           WORD *pwID,
                                           BYTE *pbyType,
                                           BYTE *pbyAttribute,
                                           BYTE *pbySecurity,
                                           char strName[18]);
```

Function description:

List first directory or file under the current directory

Parameters:

piPosition:	[out]return -1, which means that there is no file or directory to be listed. Other values means that there are files or directories which are not fully listed using this function, please use DIC_FindNextFileOrDir() to get the next directory or file.
pwID:	[out]return ID of the first directory of file
pbyType:	[out] return type of the first directory of file
pbyDirAttribute:	[out] return attribute of the first directory of file
pbyDirSecurity:	[out] return security level of the first directory of file
strName:	[out] return name of the first directory of file

Return value:

Please refer to Chapter 2 Error Code List.

1.15 List Next Directory or File under the Current Directory

```
EXTERN_C int WINAPI DIC_FindNextFileOrDir(int hic,
                                          int *piPosition,
                                          WORD *pwID,
                                          BYTE *pbyType,
                                          BYTE *pbyAttribute,
                                          BYTE *pbySecurity,
                                          char strName[18]);
```

Function descriptions:

List next directory or file under the current directory. Used together with DIC_FindFirstFileOrDir().

Parameters:

piPosition:	[out]return -1, which means that there is no files or directory to be listed. Other values means that there are files or directories which are not listed using this function, please use DIC_FindNextFileOrDir() to get the next directory or file.
pwID:	[out]return ID of the first directory of file after piPosition
pbyType:	[out] return type of the first directory of file after piPosition
pbyAttribute:	[out] return attribute of the first directory of file after piPosition
pbySecurity:	[out] return security level of the first directory of file after piPosition
strName:	[out] return name of the first directory of file after piPosition

Return value:

Please refer to Chapter 2 Error Code List.

1.16 Read Current File

```
EXTERN_C int WINAPI DIC_ReadFile(int hic, WORD wOffset, WORD *pwRead, char *pBuf);
```

Function Description:

Read the current file.

Parameters:

wOffset:	[in]specify offset of the current file to set the reading point
pwRead:	[in/out]put into the size of the data that you want to read, and with the size of data that actually get as the return value
pBuf:	[out]buffer of the read data

Note: to use this function, a file needs to be set as the current file. Please use DIC_SetCurrentFile() to set or DIC_CreateFile() will automatically create the current file.

Return value:

Please refer to Chapter 2 Error Code List.

1.17 Write the Current File

```
EXTERN_C int WINAPI DIC_WriteFile(int hic, WORD wOffset, WORD wWrite, const char *pBuf);
```

Function description:

Write the current file.

Parameters:

wOffset:	[in]specify offset of the current file to set the writing point
wWrite:	[in]put into the size of the data that you want to write
pBuf:	[in]buffer of the written data

Note: to use this function, a file needs to be set as the current file. Please use DIC_SetCurrentFile() to set or DIC_CreateFile() will automatically create the current file.

Return value:

Please refer to Chapter 2 Error Code List.

1.18 Create a Directory

```
EXTERN_C int WINAPI DIC_CreateDir(int hic,
                                  WORD wDirID,
                                  BYTE byDirType,
                                  BYTE byDirAttribute,
                                  BYTE byDirSecurity,
                                  const char strDirName[17]);
```

Function description:

Create a directory

Parameters:

wDirID:	[in]ID of the directory
byDirType:	[in] type of the directory
byDirAttribute:	[in] attribute of the directory
byDirSecurity:	[in] security level of the directory
strDirName:	[in] name of the directory

Return value:

Please refer to Chapter 2 Error Code List.

1.19 Create a File

```
EXTERN_C int WINAPI DIC_CreateFile(int hic,
                                   WORD wFileID,
                                   BYTE byFileType,
                                   BYTE byFileAttribute,
                                   WORD wFileSize,
                                   BYTE byFileSecurity,
                                   const char strFileName[18]);
```

Function description:

Create a file

Parameters:

wFileID:	[in]ID of the file
byFileType:	[in] type of the file

byFileAttribute:	[in] attribute of the file
byFileSecurity:	[in] security level of the file
strFileName:	[in] name of the file

Return value:

Please refer to Chapter 2 Error Code List.

1.20 Delete Current Directory

```
EXTERN_C int WINAPI DIC_DeleteDir(int hic);
```

Function description:

Delete current directory

Parameters:

Return value:

Please refer to Chapter 2 Error Code List.

1.21 Delete Current File

```
EXTERN_C int WINAPI DIC_DeleteFile(int hic);
```

Function description:

Delete current file

Parameters:

Return value:

Please refer to Chapter 2 Error Code List.

1.22 Get a Random Number

```
EXTERN_C int WINAPI DIC_Random(int hic, BYTE byRandomLength, BYTE abyRandom[16]);
```

Function description:

Get a random number

Parameters:

byRandomLength:	[in]length of the random number, maximum 16 bytes
abyRandom:	[out] return the random number block

Return value:

Please refer to Chapter 2 Error Code List.

1.23 Execute an Executable File

```
int WINAPI DIC_Run(int hic, WORD wRunFileNameID, WORD wParamSize, const char *cpcParam, WORD
*pwRetDataLength, char *pcRetData);
```

Function description:

Execute an executable file

Parameters:

wRunFileNameID:	[in]the executable file name, not the file ID. The name of the executable file should be in the following naming format e.g. “6AB3” “032B”, if it is “6AB3”, the parameter used as WORD will be “0x6AB3”.
wParamSize:	[in]The size of the parameter for starting-up the executable file
cpcParam:	[in]The start-up parameter. If parameters include a first 8-byte double and a second 2-byte WORD, then the input here will be a combined parameter which starts with the first double value followed by the WORD value, and wParamSize is 10.
pwRetDataLength:	[out]the length of the returned value
pcRetData:	[out]returned value. Likewise, if return values include a first 4 bytes float and a second 4 bytes DWORD, then the output will be an 8-byte number which is composed of the first 4 bytes float and the second 4 bytes DWORD, and pwRetDataLength is 8.

Return value:

Please refer to Chapter 2 Error Code List.

1.24 Create a Encrypted File

```
int WINAPI DIC_EncryptFileUseTea(int hic,
                                 WORD wFileID,
                                 BYTE byFileType,
                                 BYTE byFileAttribute,
                                 WORD *pwFileSize,
                                 BYTE byFileSecurity,
                                 const char strFileName[18],
                                 const char *cpcFileData,
                                 char *pcEncData);
```

Function description:

Use TEA algorithm to create an encrypted file and put into pcEncData. A temporary file is used during encryption, which will need to delete after this function using DIC_DeleteFile().

Parameters:

wFileID:	[in]temporary file ID
byFileType:	[in] temporary file type
byFileAttribute:	[in] temporary file attribute
pwFileSize:	[in/out]put in the size of cpcFileData and get the size of pcEncData
byFileSecurity:	[in] temporary file security level
strFileName:	[in] temporary file name
cpcFileData:	[in]the floating point to the file that needs to be encrypted
pcEncData:	[out]return the floating point of the encrypted data. Data pointed by pcEncData with size at pwFilesize can be sent to the end-user at a remote place, which can be received as the second parameter of DIC_TeaEncryptedFile2PlainFile() to do the remote update.

Return value:

Please refer to Chapter 2 Error Code List.

1.25 Decrypt the Encrypted File

```
int WINAPI DIC_TeaEncryptedFile2PlainFile(int hic, const char *cmddata);
```

Function description:

To do remote update together with DIC_EncryptFileUseTea(). To decrypt the pcEncData created using DIC_EncryptFileUseTea() back to a plain file, whose ID is the ID of the temporary file used in DIC_EncryptFileUseTea().

Parameters:

cmddata:	[in]The encrypted data pointed by pcEncData returned from DIC_EncryptFileUseTea()

Return value:

Please refer to Chapter 2 Error Code List.

1.26 Verify the Super Password

```
EXTERN_C int WINAPI DIC_CheckSuperPassword(int hic, const BYTE cabySuperPW[8]);
```

Function description:

Verify the super password

Parameters:

cabySuperPW:	[in]super password. If super password is set to zero, it will lock the dongle.

Return value:

Please refer to Chapter 2 Error Code List.

1.27 Set the Super Password

```
EXTERN_C int WINAPI DIC_SetSuperPassword(int hic, BYTE byTryTimesCeiling, const BYTE cabySuperPW[8]);
```

Function description:

Set the super password

Parameters:

byTryTimesCeiling:	[in]maximum times for attempting the super password
cabySuperPW:	[out]the super password, which should not be set to zero. Otherwise, the dongle will be locked.

Return value:

Please refer to Chapter 2 Error Code List.

1.28 Get the Remote Update Password

```
EXTERN_C int WINAPI DIC_GetRemoteUpgradePassword(int hic, DWORD dwRemoteFlag, DWORD dwHardSerial,
BYTE abyRemotePW[8]);
```

Function description:

Get the remote update password.

Parameters:

dwRemoteFlag:	[in]the remote update identity
dwHardSerial:	[in]the hardware serial number for this particular remote update
abyRemotePW:	[out]input the remote update password for this time and get the next remote update password

Return value:

Please refer to Chapter 2 Error Code List.

1.29 Format the Dongle

```
EXTERN_C int WINAPI DIC_FormatCard(int hic, char strVolume[17], char strManufactureInfo[16]);
```

Function description:

Format the dongle memory

Parameters:

strVolume:	[in]volume
strManufactureInfo:	[in]manufacturer information

Note: formatting will only take effective after re-plugging of the dongle

Return value:

Please refer to Chapter 2 Error Code List.

1.30 Encrypt a block of Data Using DES Algorithm

```
EXTERN_C int WINAPI DIC_DesEncrypt(int hic, WORD wKeyFileID, BYTE byKeyIndex, WORD wDataLength, const char *cpcDataSrc, char *pcDataEnc);
```

Function description:

Encrypt a block of data using DES algorithm

Parameters:

wKeyFileID:	[in]encryption key file
byKeyIndex:	[in]index of the key inside the key file
wDataLength:	[in]length of the data block
cpcDataSrc:	[in]data block
pcDataEnc:	[out]encrypted data

Note: for DES algorithm, length of the data block must be multiples of the length of the key.

Return value:

Please refer to Chapter 2 Error Code List.

1.31 Decrypt Using DES Algorithm

```
EXTERN_C int WINAPI DIC_DesDecrypt(int hic, WORD wKeyFileID, BYTE byKeyIndex, WORD wDataLength, const char *cpcDataSrc, char *pcDataDec);
```

Function description:

Decryption of DES algorithm

Parameters:

wKeyFileID:	[in]ID of key file

byKeyIndex:	[in]index of the key, please refer to DIC_GenerateDesKeyFile() description
wDataLength:	[in]length of the data to be decrypted
cpcDataSrc:	[in]data block to be decrypted
pcDataDec:	[out]plain data after decryption

Return value:

Please refer to Chapter 2 Error Code List.

1.32 **Create a DES key file**

```
EXTERN_C int WINAPI DIC_GenerateDesKeyFile(int hic, WORD wKeyFileID, WORD wKeyDataLength, const char *cpcKeyData);
```

Function description:

Generate a DES key file, either 8 bytes or 16 bytes long. If the key file is 16 bytes long, 3DES algorithm will automatically be used.

Parameters:

wKeyFileID:	[in]ID of the key file
wKeyDataLength:	[in]Key length
cpcKeyData:	[in]Key data

Note: normally, key data is in the following format:
"\x03\x10\x01\x02\x03\x04\x05\x06\x07\x08\x09\x0A\x0B\x0C\x0D\x0E\x0F\x10\x01\x08\x01\x02\x03\x04\x05\x06\x07\x08"
The first “\x03” means that index of the following key is 3; the second “\x10 (hexadecimal)” means that it is a 16-byte key and the following 16 bytes is the key itself. The 19th byte in the above key data, “\x01”, means that index of the following key is 1; the followed “\x08” means that it is a 8-byte key and the following 8 bytes is the key. It is easy to use DIC_WriteFile() to append more keys in the key data.

E.g. in the above key data, a new key with index 2, 8-byte long will be appended:

```
DIC_GenerateDesKeyFile(hic, 0x1003, 28,
"\x03\x10\x01\x02\x03\x04\x05\x06\x07\x08\x09\x0A\x0B\x0C\x0D\x0E\x0F\x10\x01\x08\x01\x02\x03\x04\x05\x06\x07\x08");
DIC_SetCurrentFile(hic, 0x1003);
DIC_WriteFile(hic, 28, 10, "\x02\x08\x01\x02\x03\x04\x05\x06\x07\x08");
```

Return value:

Please refer to Chapter 2 Error Code List.

1.33 Generate RSA Key Pair

```
EXTERN_C int WINAPI DIC_GenerateRsaKey(int hic, WORD wPublicKeyFileID, WORD wPublicKeySize, WORD wPrivateKeyFileID);
```

Function description:

Generate a RSA key pair based on ID and size. Currently only two sizes, 512 and 1024 are available.

Parameters:

wPublicKeyFileID:	[in]public key ID
wPublicKeySize:	[in]size of the public key
wPrivateKeyFileID:	[in]private key ID

Return value:

Please refer to Chapter 2 Error Code List.

1.34 Encrypt a Block of Data Using RSA Algorithm

```
EXTERN_C int WINAPI DIC_RsaEncrypt(int hic, WORD wPublicKeyFileID, WORD wDataLength, const char *cpcDataSrc, char *pcDataEnc, BOOL bIsDataSrcFromOpenssl);
```

Function description:

Encrypt a block of data using RSA algorithm.

Parameters:

wPublicKeyFileID:	[in]public key ID
wDataLength:	in]length of the data block
cpcDataSrc:	[in]data block
pcDataEnc:	[out]encrypted data
bIsDataSrcFromOpenssl:	[in]whether the data block is sourced from OpenSSL (data outputted from OpenSSL needs to be reversed before putting into the dongle)

Note: RSA algorithm: plain data encrypted by private key will need to be decrypted by the paired public key, and vice versa.
The length of data block shall be multiples of 512/8 = 64 or 1024/8 = 128.

Return value:

Please refer to Chapter 2 Error Code List.

1.35 Decrypt Data Using RSA Algorithm

```
EXTERN_C int WINAPI DIC_RsaDecrypt(int hic, WORD wPrivateKeyFileID, WORD wDataLength, const char
*cpcDataSrc, char *pcDataDec, BOOL bIsDataSrcFromOpenssl);
```

Function description:

Decrypt data using RSA algorithm.

Parameters:

wPrivateKeyFileID:	[in]private key file
wDataLength:	[in]length of the encrypted data
cpcDataSrc:	[in]the encrypted data
pcDataDec:	[out]plain data
bIsDataSrcFromOpenssl:	[in] whether the data block is sourced from OpenSSL (data outputted from OpenSSL needs to be reversed before inputting into the dongle)

Return value:

Please refer to Chapter 2 Error Code List.

1.36 Get Size of the Free Space on the Dongle

```
EXTERN_C int WINAPI DIC_GetFreeSpace(int hic, DWORD *pdwFreeSpace);
```

Function description:

Get size of the free space on the dongle

Parameters:

pdwFreeSpace:	[out]return size of the free space

Return value:

Please refer to Chapter 2 Error Code List.

1.37 Set Initial Value of the Counter

```
EXTERN_C int WINAPI DIC_SetCounter(int hic, DWORD dwInit);
```

Function descriptions:

Set the initial value of the counter

Parameters:

dwInit:	[in]initial value of the counter

Return value:

Please refer to Chapter 2 Error Code List.

Chapter 2. ROCKEY6SMART Error Code List

2.1 Constant Error Code

No.	Error Code	Value	Description
1	SCARD_S_SUCCESS	0x00000000L	Success, no error.
2	SCARD_F_INTERNAL_ERROR	0x80100001L	Internal connection error
3	SCARD_E_CANCELLED	0x80100002L	Action canceled by user
4	SCARD_E_INVALID_HANDLE	0x80100003L	Invalid handle
5	SCARD_E_INVALID_PARAMETER	0x80100004L	Invalid parameters (P1,P2)
6	SCARD_E_INVALID_TARGET	0x80100005L	Invalid starting information
7	SCARD_E_NO_MEMORY	0x80100006L	Not enough memory
8	SCARD_F_WAITED_TOO_LONG	0x80100007L	Waited too long
9	SCARD_E_INSUFFICIENT_BUFFER	0x80100008L	Insufficient buffer
10	SCARD_E_UNKNOWN_READER	0x80100009L	Unknown reader
11	SCARD_E_TIMEOUT	0x8010000AL	Time out
12	SCARD_E_SHARING_VIOLATION	0x8010000BL	Sharing violation
13	SCARD_E_NO_SMARTCARD	0x8010000CL	No smart card
14	SCARD_E_UNKNOWN_CARD	0x8010000DL	Unknown card
15	SCARD_E_CANT_DISPOSE	0x8010000EL	Cannot dispose card
16	SCARD_E_PROTO_MISMATCH	0x8010000FL	Communication protocol mismatch
17	SCARD_E_NOT_READY	0x80100010L	Card not ready
18	SCARD_E_INVALID_VALUE	0x80100011L	Invalid values
19	SCARD_E_SYSTEM_CANCELLED	0x80100012L	Action canceled by system
20	SCARD_F_COMM_ERROR	0x80100013L	Internal communication error
21	SCARD_F_UNKNOWN_ERROR	0x80100014L	Internal unknown error
22	SCARD_E_INVALID_ATR	0x80100015L	Invalid ATR
23	SCARD_E_NOT_TRANSACTED	0x80100016L	Trying to stop nonexistent process
24	SCARD_E_READER_UNAVAILABLE	0x80100017L	Reader unavailable
25	SCARD_P_SHUTDOWN	0x80100018L	Process stopped, shut-down allowed
26	SCARD_E_PCI_TOO_SMALL	0x80100019L	PCI too small
27	SCARD_E_READER_UNSUPPORTED	0x8010001AL	Unsupported reader
28	SCARD_E_DUPLICATE_READER	0x8010001BL	Reader name already exist
29	SCARD_E_CARD_UNSUPPORTED	0x8010001CL	Card not supported by Reader
30	SCARD_E_NO_SERVICE	0x8010001DL	No smart card service
31	SCARD_E_SERVICE_STOPPED	0x8010001EL	Smart card service stopped
32	SCARD_E_UNEXPECTED	0x8010001FL	Unexpected smart card error
33	SCARD_E_ICC_INSTALLATION	0x80100020L	Cannot get smart card supplier info
34	SCARD_E_ICC_CREATEORDER	0x80100021L	Cannot get smart card manufacturer info
35	SCARD_E_UNSUPPORTED_FEATURE	0x80100022L	Unsupported features

36	SCARD_E_DIR_NOT_FOUND	0x80100023L	Directory not found
37	SCARD_E_FILE_NOT_FOUND	0x80100024L	File not found
38	SCARD_E_NO_DIR	0x80100025L	Not effective directory
39	SCARD_E_NO_FILE	0x80100026L	Nonexistent file
40	SCARD_E_NO_ACCESS	0x80100027L	File no access
41	SCARD_E_WRITE_TOO_MANY	0x80100028L	No free space
42	SCARD_E_BAD_SEEK	0x80100029L	Pointer error
43	SCARD_E_INVALID_CHV	0x8010002AL	PIN Error
44	SCARD_E_UNKNOWN_RES_MNG	0x8010002BL	Unknown error from smart card
45	SCARD_E_NO_SUCH_CERTIFICATE	0x8010002CL	Nonexistence certificate
46	SCARD_E_CERTIFICATE_UNAVAILABLE	0x8010002DL	Certificate unavailable
47	SCARD_E_NO_READERS_AVAILABLE	0x8010002EL	No reader available
48	SCARD_E_COMM_DATA_LOST	0x8010002FL	Data lost during communication
49	SCARD_E_NO_KEY_CONTAINER	0x80100030L	No key container
50	SCARD_W_UNSUPPORTED_CARD	0x80100065L	ATR conflicts
51	SCARD_W_UNRESPONSIVE_CARD	0x80100066L	Unresponsive card
52	SCARD_W_UNPOWERED_CARD	0x80100067L	Unpowered card
53	SCARD_W_RESET_CARD	0x80100068L	Card reset
54	SCARD_W_REMOVED_CARD	0x80100069L	Card removed
55	SCARD_W_SECURITY_VIOLATION	0x8010006AL	Access denied
56	SCARD_W_WRONG_CHV	0x8010006BL	Access denied without PIN
57	SCARD_W_CHV_BLOCKED	0x8010006CL	PIN maximum attempts reached
58	SCARD_W_EOF	0x8010006DL	End of file
59	SCARD_W_CANCELLED_BY_USER	0x8010006EL	Operation cancelled by user
60	SCARD_W_CARD_NOT_AUTHENTICATED	0x8010006FL	PIN not set yet

2.2 Extended Error Code

No	Error Code	Value	Description
1	SCARD_E_FILE_EXISTS	0xA0100001L	File already exists
2	SCARD_E_EPROM_ERROR	0xA0100002L	EPRROM error
3	SCARD_E_INVALID_CLA	0xA0100003L	Invalid CLA
4	SCARD_E_INVALID_INS	0xA0100004L	Invalid INS
5	SCARD_E_VM_ADDRESS_ERROR	0xA0100005L	VM address over bound/error
6	SCARD_E_ZERO_DIVIDE	0xA0100006L	Divide by zero error
7	SCARD_E_WRONG_POSITION	0xA0100007L	Card wrong position
8	SCARD_E_UNKNOWN_STATE	0xA0100008L	Card unknown state
9	SCARD_E_CARD_NOT_OPENED	0xA0100009L	Card not opened
10	SCARD_E_UNKNOWN_COMMAND	0xA010000AL	Unknown command
11	SCARD_E_ZERO_TRYTIME	0xA010000BL	Super password PIN attempt time

			set to zero
12	SCARD_E_TOO_MANY_DEVICE	0xA010000CL	Too many devices
13	SCARD_E_INVALID_INSTRUCTION	0xA010000DL	Invalid instructions
14	SCARD_W_RESPONSE	0xA01000FFL	More return values
15	RETURN_LENGTH	0xA0100014L	Return length error, more return values

2.3 Virtual Device Exclusive Error Code

No	Error Code	Value	Description
1	SCARD_E_FILE_CREATE_FAILED	0xA0101001L	Fail to create a file
2	SCARD_E_FILE_OPEN_FAILED	0xA0101002L	Fail to open a file
3	SCARD_E_FLOAT_NOT_FOUND	0x0000e00aL	Cannot find FLOAT
4	SCARD_E_RSADES_NOT_FOUND	0x0000e00bL	Cannot find RSA_DES
5	SCARD_E_LIBS_ERROR	0x0000e00cL	Library error
6	SCARD_E_EXTENDFLOAT_NOT_FOUND	0x0000e00dL	Cannot find extended FLOAT

2.4 IO Error

No	Error Code	Value	Description
1	ROCKEY_K_NOT_FOUND	0xF0100001L	Cannot find dongle
2	ROCKEY_K_QUERY_FAILED	0xF0100002L	Dongle query failed
3	ROCKEY_K_WRITE_FAILED	0xF0100003L	Writing dongle failed
4	ROCKEY_K_READ_FAILED	0xF0100004L	Reading dongle failed

Chapter 3. Appendix

3.1 Additional Explanation

3.1.1 Directory/File ID (2 bytes)

Each file or directory inside the dongle has an ID which is a 16-bit number. It is necessary to use different IDs to separate sub-directories and files under one directory, which are used as the unique identifications for the files or the directories.

Note:

3F00 is the ID for root directory;

2F01 is the ID for ATR file;

3FFF is the ID for the current file (not used);

0000 (system reserved)

Please do not use the above IDs when creating a file or a directory.

3.1.2 Directory/File Type (1 byte)

Each file or directory has a type description code, which is called as a FCLA code by the system to differentiate a file type. Files that have the same type code will be considered as same type files.

The executable file inside the dongle is assigned a FCLA code, and can only create and write/read the same type (same FCLA code) data files.

FCLA code takes one byte in the IC card, ranging from 0 to FF.

FF, in the system, is considered as “no type”, so that data files with “FF” type can be visited by any type of executable files.

“System current file type” will be set to the file type of the executable file when “file select” function is used to visit the executable file in the dongle. (SYSFCLA=FCLA):

If the executable has file type “FF”, SYSFCLA will remain unchanged.

If the file type is set to the new one, “system current file security level” will be set to zero. (SYSPRI = 0).

After reset, SYSFCLA of the dongle, by default, is FF.

System root directory has type FF.

In the dongle, same type of files creates an independent sub system. On the contrary, all data files with type FF are public data, and all executable files with type FF are public executables.

Public executable files can only read/write public data files.

Notes to developers:

File type can be used in management to manage many products or in the case that many developers are involved to develop one product.

It is suggested that all file type to be set to FF if only one set of software is protected by the dongle.

It is also suggested to set the same file type for files and directories under the same directory and use "security level" to control read/write.

3.1.3 Directory/File Privilege (Security Level) Code (1/2 byte)

Except type, each file or directory has a privilege (or security level), which can be used to limit or control read/write privilege of the same type of files.

File privilege (FPRI) is designed for the same types of file, so that it is one level lower than FCLA.

FPRI takes half byte on the IC card, ranging from 0-15. There are 16 different privileges. 0 is the lowest and 15 is the highest.

The dongle system uses one byte to record the current file's FPRI as the SYSFPRI, the value of which can only be increased by executing the executable file in the dongle. However, the SYSFPRI cannot exceed the FPRI of the executable file (SYSFPRI <= FPRI).

The executable file can only operate lower security level data files (read/write or create). The executable file, during exiting, can clear SYSFPRI (SYSFPRI = 0).

For an executable file, there is a file attribute "FILEATTR_UPIGNORE", the setting of which tells the system that if SYSFPRI >= the FPRI of the executable, there is no need to call the executable any more.

Notes to developers:

File privilege provides a higher degree of management on files. If not needed, all file privilege can be set to 0.

It is necessary to consider either FCLA or FPRI in specific case. FPRI can only be used as there is no difference in FCLA.

3.1.4 Directory/File Attribute (1/2 byte)

Each file or directory has an attribute (FATTR):

FILEATTR_NORMAL	0x00// Normal file	a)
FILEATTR_EXEC	0x10// Executable file	b)
FILEATTR_DIR	0x20// It is a directory, not a file	c)
FILEATTR_UPIGNORE	0x40// File privilege higher than current, ignore	d)
FILEATTR_INTERNAL	0x80// Internal file	e)

a) Normal file.

b) Executable attribute: it is an executable file inside the dongle, which can only be put under the root directory and can only be created or deleted after the super password is verified.

c) Directory attribute: it is a directory instead of a file. Once set, the other attribute will be ignored.

d) Ignoring attribute: It is only effective for executable files inside the dongle. It is used in the case that when SYSFPRI >= FPRI (system file privilege is larger than the privilege of the executable file), the executable file will be ignored.

e) Internal attribute: internal files can only be operated by the executable file onboard the dongle or be deleted after the super password is verified. No other operation is allowed.

3.1.5 Directory/File Name

When a file or a directory is created, a name less than 16 character long can be given. Once the name is set, operation on the file can be designated either by the file ID or the file name.

3.2 .Explanation on Other Functions

Format operation can only be done after the super password is verified. During and only during formatting, the volume and the manufacturer info can be updated.

It is necessary to provide directory ID, type, security level and name when a directory is created.

Directory listing function is based on enumeration until a “-1” is returned, which means that there are no

more directories or files to be listed.

DIC_SetCurrentDir() is similar to DOS command “cd …” which only needs the directory ID or directory name. If the directory name is used, it is necessary to set the directory ID as 0.

During creation of a file, the newly created file will be set as the current file. There is no need for DIC_SetCurrentFile().

3.3 Explanation on Remote Update

1. The idea of introducing remote update tag (the RemoteTag) and password (the RemotePass) is to realize “one password for one update” concept so as to provide safe remote update to the dongles.
2. The whole remote update procedure is done by the help of both the end-user and the software provider.

   Detailed steps are:

   a. Creation of an initial remote update password

   This step is done at the software provider side as the initial work before the dongle is sent to the end-user. Once the initial remote update password is created, it cannot be changed anymore unless the software provider uses the super password to format the dongle.

   b. The current remote update information

   c. The hardware ID

   When the end-user wants to remote update the dongle at hand, it is necessary for the end-user to use the remote update tool GUI to read the current remote update information (initial remote update tag and password) and hardware ID, and then send these information to the software provider through certain ways.

   d. Upgrade to the super user

   e. Creation of a new remote update password

   The software provider, as a super user, produce a new remote update password and remote update file for the dongle to be sent to the end-user.

   f. Verification of the new remote update password

   g. Update

The end-user, upon receiving the new remote update password, will input the new password on the remote update tool GUI to verify. If verification passes, the remote update tag (the RemoteTag) and initial password will be reset. The RemoteTag can be checked by the executable files on board the dongle. It is suggested to use the executable files on board the dongle to check the RemoteTag (enhanced security) to finalize the remote update procedure.

3. About the remote update tag – RemoteTag. Please note that it is not the remote update password, but used by the software provider to identify the update status, e.g. software version can be used as the RemoteTag. The RemoteTag is a 32-bit number. The highest bit (0x80000000) is used to identity whether the new remote update password is related to the hardware ID. If the highest bit is 1, it means related; otherwise, if the highest bit is 0, the new remote update password is only related to the previous remote update password, not the hardware ID. It is to simplify the remote update maintenance work for the software provider.
4. Additional explanation: the RemoteTag and the RemotePass provided by the software provider are related, neither of which can be changed by the end-user. If the highest bit of the RemoteTag is 1, the new RemotePass is only effective to the specific hardware. It is necessary for the end-user to provide the RemotePass and the hardware ID at least for the remote update procedure. The RemoteTag can be used as reference for the software provider, which will have no effect on the next remote update procedure. However, a new RemoteTag shall be set in order to identify a new update status.