

Rockey6Smart 升级新增 网络锁 API 接口手册



版权所有© 2009, 北京飞天诚信科技有限公司
<http://www.FTsafe.com>

北京飞天诚信科技有限公司（以下简称“飞天公司”）尽最大努力使这篇文章中的内容完善且正确。飞天公司对于由这篇文档导致的任何形式的直接或间接损失不负有责任。这篇文章的内容会跟随产品的升级而有所变化。

目 录

目 录	3
一、Rockey6Smart升级新增 网络锁API接口	5
1.1 获取卷标和开发商信息	5
1.2 获取生产时间、序列号、销售时间、COS版本信息	5
1.3 获取区域编码、代理商编码和两个用户编码	6
1.4 取得密码验证状态	6
1.5 获取远程升级标志和升级密码	7
1.6 设置远程升级标志和升级密码	7
1.7 验证远程升级标志和升级密码是否正确	7
1.8 获取当前目录ID、类别、属性、安全级别和目录名	8
1.9 设置当前目录	8
1.10 获取当前文件ID、类别、属性、安全级别和文件名	9
1.11 设置当前文件	9
1.12 获取上级目录ID、类别、属性、安全级别和目录名	10
1.13 回到上级目录	10
1.14 列出当前目录下的所有目录和文件	10
1.15 列出当前目录下的所有目录和文件	11
1.16 读取当前文件	12
1.17 写当前文件	12
1.18 创建一个目录	13
1.19 创建一个文件	13
1.20 删除当前目录	14
1.21 删除当前文件	14
1.22 取得一个随机数	14
1.23 运行一个可执行文件	15
1.24 采用DES加密算法加密一段数据	15
1.25 采用DES算法解密	16
1.26 生成一个DES密钥文件	16
1.27 采用RSA算法加密一段数据	17
1.28 采用RSA算法解密一段数据	17
1.29 获取加密锁的剩余空间大小	18
1.30 使计数器值减 1	18
1.31 使用Openssl生成一个公钥文件	18
1.32 使用Openssl生成一个私钥文件	19
1.33 锁定网络锁	19
1.34 解锁网络锁	20
1.35 获取当前路径	20
二、Rockey6Smart 错误码定义	21
2.1 错误码常量列表	21
2.2 虚拟设备专用错误码	23

2.3 网络锁专用代码.....	23
2.4 网络错误码.....	24
2.5 扩展错误码.....	25
2.6 NetBios扩展错误码参考.....	26
三、附录.....	28
3.1 有关文件系统的说明.....	28
3.2 其它功能的说明.....	30

一、Rockey6Smart 升级新增 网络锁 API 接口

1.1 获取卷标和开发商信息

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

功能说明:

获取卷标和开发商信息

参数说明:

strVolume: [out]返回加密锁的卷标
strManufactureInfo: [out]返回加密锁内的开发商信息
这两个参数开发商可以更改，最终用户不可以更改

返回值:

见后面的返回值错误码列表

1.2 获取生产时间、序列号、销售时间、COS 版本信息

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

功能说明:

获取生产时间、序列号、销售时间、COS 版本信息

参数说明:

pdwProduceTime: [out]返回加密锁的生产时间
pdwSerial: [out]返回加密锁的序列号
pdwSalesTime: [out]返回加密锁的销售时间
pdwCOSVersion: [out]返回加密锁内的 COS 版本
这四个参数出厂时由生产商设定，不可更改

返回值:

见后面的返回值错误码列表

1.3 获取区域编码、代理商编码和两个用户编码

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

功能说明：

获取区域编码、代理商编码和两个用户编码

参数说明：

pwZoneCode: [out]返回区域编码
pwAgentCode: [out]返回代理商编码
pwUser1Code: [out]返回用户的编码
pwUser2Code: [out]返回用户的编码

返回值：

见后面的返回值错误码列表

1.4 取得密码验证状态

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

功能说明：

取得密码验证状态，由变量 piVerifyState 返回

参数说明：

piVerifyState: [out] 密码验证状态，有如下几个值：

表一

Flag	意义
NO_PASSED	密码没有验证通过
SUPER_PASSWORD_PASSED	超级密码验证通过
REMOTE_PASSWORD_PASSED	远程升级密码验证通过
SUPER_REMOTE_PASSWORD_PASSED	超级密码和远程升级密码均验证通过

这些值的定义见后面的常量定义一节

返回值：

见后面的返回值错误码列表

1.5 获取远程升级标志和升级密码

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

功能说明:

获取远程升级标志和升级密码。

参数说明:

pdwFlag:	[out]返回远程升级标志
abyPassword:	[out]返回远程升级密码

返回值:

见后面的返回值错误码列表

1.6 设置远程升级标志和升级密码

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

功能说明:

设置远程升级标志和升级密码

参数说明:

dwFlag:	[in]设置远程升级标志
abyPassword:	[in]设置远程升级密码

返回值:

见后面的返回值错误码列表

1.7 验证远程升级标志和升级密码是否正确

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

功能说明:

验证远程升级标志和升级密码是否正确

参数说明:

dwFlag:	[in]要验证的远程升级标志
dwPassword:	[in]要验证的远程升级密码

返回值:

见后面的返回值错误码列表

1.8 获取当前目录 ID、类别、属性、安全级别和目录名

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

功能说明:

获取当前目录 ID、类别、属性、安全级别和目录名

参数说明:

pwDirID:	[out]返回当前目录 ID
pbyDirType:	[out]返回当前目录类别
pbyDirAttribute:	[out]返回当前目录属性
pbyDirSecurity:	[out]返回当前目录安全级别
strDirName:	[out]返回当前目录名

返回值:

见后面的返回值错误码列表

1.9 设置当前目录

```
EXTERN_C int WINAPI DICNet_SetCurrentDir(int hic,
                                         WORD wDirID,
                                         char strDirName[17]);
```

功能说明:

设置当前目录为 wDirID 标识的目录

参数说明:

wDirID:	[in]目录 ID
strDirName:	[in]目录名

备注:

可以通过目录ID或目录名设置当前目录，传入其中一个值即可，另一个值可以设为0或NULL，两个值都传入时这两个值应该代表同一个目录

返回值:

见后面的返回值错误码列表

1.10 获取当前文件 ID、类别、属性、安全级别和文件名

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

功能说明:

获取当前文件 ID、类别、属性、安全级别和文件名

参数说明:

pwFileID:	[out]返回当前文件 ID
pbyFileType:	[out]返回当前文件类别
pbyFileAttribute:	[out]返回当前文件属性
pbyFileSecurity:	[out]返回当前文件安全级别
strFileName:	[out]返回当前文件名

返回值:

见后面的返回值错误码列表

1.11 设置当前文件

```
EXTERN_C int WINAPI DICNet_SetCurrentFile(int hic,
                                           WORD wFileID,
                                           char strFileName[18]);
```

功能说明:

设置当前文件为 wFileID 标识的文件

参数说明:

wFileID:	[in]文件 ID
strFileName:	[in]文件名

备注:

可以通过文件ID或文件名设置当前文件，传入其中一个值即可，另一个值可以设为0或NULL，两个值都传入时这两个值应该代表同一个文件

返回值:

见后面的返回值错误码列表

1.12 获取上级目录 ID、类别、属性、安全级别和目录名

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

功能说明:

获取上级目录 ID、类别、属性、安全级别和目录名

参数说明:

pwDirID:	[out]返回上级目录 ID
pbyDirType:	[out]返回上级目录类别
pbyDirAttribute:	[out]返回上级目录属性
pbyDirSecurity:	[out]返回上级目录安全级别
strDirName:	[out]返回上级目录名

返回值:

见后面的返回值错误码列表

1.13 回到上级目录

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

功能说明:

回到上级目录

参数说明:

返回值:

见后面的返回值错误码列表

1.14 列出当前目录下的所有目录和文件

```
EXTERN_C int WINAPI DICNet_FindFirstFileOrDir(int hic,
                                                int *piPosition,
                                                WORD *pwID,
                                                BYTE *pbyType,
                                                BYTE *pbyAttribute,
```

```
BYTE *pbySecurity,  
char strName[18]);
```

功能说明:

列出当前目录下的所有目录和文件，与 DIC_FindNextFileOrDir 配合使用

参数说明:

piPosition: [out]返回-1 表示当前目录下没有目录或文件，其它表示还有目录或文件没有列出，可以通过调用 DIC_FindNextFileOrDir 函数获取剩余目录或文件

pwID: [out]返回当前目录下第个目录或文件的 ID

pbyType: [out]返回当前目录下第个目录或文件的类别

pbyDirAttribute: [out]返回当前目录下第个目录或文件的属性

pbyDirSecurity: [out]返回当前目录下第个目录或文件的安全级别

strName: [out]返回当前目录下第个目录或文件的名称

返回值:

见后面的返回值错误码列表

1.15 列出当前目录下的所有目录和文件

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

功能说明:

列出当前目录下的所有目录和文件，与 DIC_FindFirstFileOrDir 配合使用

参数说明:

piPosition: [out]返回-1 表示已经列出当前目录下的所有目录和文件，其它表示还有目录或文件没有列出，可以通过继续调用 DIC_FindNextFileOrDir 函数获取剩余目录或文件

pwID: [out]返回当前目录下 piPosition 后的下个目录或文件的 ID

pbyType: [out]返回当前目录下 piPosition 后的下个目录或文件的类别

pbyDirAttribute: [out]返回当前目录下 piPosition 后的下个目录或文件的属性

pbyDirSecurity: [out]返回当前目录下 piPosition 后的下个目录或文件的安全级别

strName: [out]返回当前目录下 piPosition 后的下个目录或文件的名称

返回值:

见后面的返回值错误码列表

1.16 读取当前文件

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

功能说明:

读取当前文件

参数说明:

wOffset:	[in]当前文件偏移量
pwRead:	[in out]传入要读取的数据大小, 返回实际读取的数据大小
pBuf:	[out]接收数据的缓冲区

备注:

读写文件时要首先将文件设置为当前文件, 可通过DIC_SetCurrentFile设置, DIC_CreateFile会自动将创建的文件设置为当前文件

返回值:

见后面的返回值错误码列表

1.17 写当前文件

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

功能说明:

写当前文件

参数说明:

wOffset:	[in]当前文件偏移量
wWrite:	[in]要写入的数据大小
pBuf:	[in]写入数据存放的缓冲区

备注:

读写文件时要首先将文件设置为当前文件, 可通过DIC_SetCurrentFile设置, DIC_CreateFile会自动将创建的文件设置为当前文件

返回值:

见后面的返回值错误码列表

1.18 创建一个目录

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

功能说明:

创建一个目录

参数说明:

wDirID:	[in]要创建的目录 ID
byDirType:	[in]创建的目录类别
byDirAttribute:	[in]创建的目录属性
byDirSecurity:	[in]创建的目录安全级别
strDirName:	[in]创建的目录名

返回值:

见后面的返回值错误码列表

1.19 创建一个文件

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

功能说明:

创建一个文件

参数说明:

wFileID:	[in]要创建的文件 ID
byFileType:	[in]创建的文件类别
byFileAttribute:	[in]创建的文件属性
byFileSecurity:	[in]创建的文件安全级别
strFileName:	[in]创建的文件名

返回值:

见后面的返回值错误码列表

1.20 删除当前目录

EXTERN_C int WINAPI DICNet_DeleteDir(int hic);

功能说明:

删除当前目录

参数说明:

返回值:

见后面的返回值错误码列表

1.21 删除当前文件

EXTERN_C int WINAPI DICNet_DeleteFile(int hic);

功能说明:

删除当前文件

参数说明:

返回值:

见后面的返回值错误码列表

1.22 取得一个随机数

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

功能说明:

取得一个随机数

参数说明:

byRandomLength: [in]随机数长度，最大为 16

abyRandom: [out]存放随机数的数组

返回值:

见后面的返回值错误码列表

1.23 运行一个可执行文件

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

功能说明:

运行一个可执行文件

参数说明:

wRunFileNameID: [in]可执行文件的文件名表示的文件 ID, 注意不是可执行文件的文件 ID, 可执行文件的文件名必须是如下形式“6AB3”、“032B”, 即一个用字符串表示的 WORD 值,例如可执行文件的文件名为“6AB3”, 则该参数应传入 0x6AB3

wParamSize: [in]传入启动时的参数大小

cpcParam: [in]传入的启动参数。启动参数是将各个输入参数连接成的参数, 例如输入参数为一个 8 字节的 double 和一个 2 字节的 WORD 则 cpcParam 的前个字节是这个输入的 double 值, 后个字节是输入的 WORD 值, wParamSize 的大小为 10

pwRetDataLength: [out]执行后的返回数据的大小

pcRetData: [out]执行后的返回数据。这个参数由程序执行后的返回数据连接而成, 例如如果返回数据包括一个 4 字节的 float 和一个 4 字节的 DWORD 则该参数的前 4 个字节是这个返回的 float 值, 后 4 个字节是 DWORD 值, *pwRetDataLength 的大小为 8

返回值:

见后面的返回值错误码列表

1.24 采用 DES 加密算法加密一段数据

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

功能说明:

采用 DES 加密算法加密一段数据

参数说明:

wKeyFileID: [in]密钥文件 ID

byKeyIndex: [in]密钥文件中的密钥索引, 即密钥文件中的第多少个密钥

wDataLength: [in]要加密的数据长度

cpcDataSrc: [in]要加密的数据

pcDataEnc: [out]加密后的数据

备注:

对于DES算法, 加解密数据的长度必须是密钥长度个字节的倍数

返回值:

见后面的返回值错误码列表

1.25 采用 DES 算法解密

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

功能说明:

采用 DES 算法解密

参数说明:

wKeyFileID:	[in]密钥文件 ID
byKeyIndex:	[in]密钥文件中的密钥索引, 见 DIC_GenerateDesKeyFile 函数说明中有关密钥索引的说明
wDataLength:	[in]要解密的数据长度
cpcDataSrc:	[in]要解密的数据
pcDataDec:	[out]解密后的数据

返回值:

见后面的返回值错误码列表

1.26 生成一个 DES 密钥文件

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

功能说明:

生成一个 DES 密钥文件, 密钥长度只有 8 个字节和 16 个字节两种, 如果是 16 个字节的密钥, 加解密时会自动采用 3DES 算法

参数说明:

wKeyFileID:	[in]生成的密钥文件 ID
wKeyDataLength:	[in]密钥数据的长度
cpcKeyData:	[in]密钥数据

备注:

密钥数据是如下形式的数据:

"\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" 其中第一个\x03表示后面紧接的是索引为3的密钥, 第二个\x10(16进制表示)表示这个密钥的长度是16个字节,紧接的16个字节是这个索引为3的密钥。接下来的第19个字节\x01表示后面紧接的是索引为1的密钥, \x08表示这个密钥的长度是8个

字节紧接的8个字节是这个索引为1的密钥。用户也可以通过调用DIC_WriteFile函数往该密钥文件中继续写入密钥。

例如要先生成一个包含上述密钥数据的DES密钥文件，接着写入一条索引为2，长度为8个字节的密钥，代码如下：

```
DIC_GenerateDesKeyFile(hic, 0x1003, 28,  
"x03x10x01x02x03x04x05x06x07x08x09x0Ax0Bx0Cx0Dx0Ex0Fx10x01x08x01x  
02x03x04x05x06x07x08");  
DIC_SetCurrentFile(hic, 0x1003);  
DIC_WriteFile(hic, 28, 10, "x02x08x01x02x03x04x05x06x07x08");
```

返回值：

见后面的返回值错误码列表

1.27 采用 RSA 算法加密一段数据

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

功能说明：

采用 RSA 算法加密一段数据

参数说明：

wPublicKeyFileID:	[in]公钥文件 ID
wDataLength:	[in]加密的数据长度
cpcDataSrc:	[in]加密的数据
pcDataEnc:	[out]加密后的数据
bIsDataSrcFromOpenssl:	[in]加密的数据是否来源于 Openssl 操作过的数据（因为 Openssl 的字节序需要在传入锁内前翻转）

备注：

对于RSA算法，对一段明文先用私钥解密再用公钥加密＝先用公钥加密再用私钥解密＝明文，加解密数据的长度必须是密钥长度512/8=64或1024/8=128字节的倍数

返回值：

见后面的返回值错误码列表

1.28 采用 RSA 算法解密一段数据

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

功能说明:

采用 RSA 算法解密一段数据

参数说明:

wPrivateKeyFileID: [in]私钥文件 ID
 wDataLength: [in]解密的数据长度
 cpcDataSrc: [in]解密的数据
 pcDataDec: [out]解密后的数据
 bIsDataSrcFromOpenssl: [in]解密的数据是否来源于 Openssl 操作过的数据（因为 Openssl 的字节序需要在传入锁内前翻转）

返回值:

见后面的返回值错误码列表

1.29 获取加密锁的剩余空间大小

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

功能说明:

获取加密锁的剩余空间大小

参数说明:

pdwFreeSpace: [out]返回加密锁的剩余空间大小

返回值:

见后面的返回值错误码列表

1.30 使计数器值减 1

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

功能说明:

使计数器值减 1

参数说明:

返回值:

见后面的返回值错误码列表

1.31 使用 Openssl 生成一个公钥文件

```
int WINAPI DICNet_GenerateRsaPublicKeyFile(int hic, WORD wPublicKeyFileID, WORD
```

```
wPublicKeySize, const BYTE *cpbyPublicKeyData);
```

功能说明:

使用 Openssl 生成一个公钥文件

参数说明:

wPublicKeyFileID	[in]公钥文件 ID
wPublicKeySize	[in]公钥大小
cpbyPublicKeyData	[in]公钥数据, 一般为用 BN_bn2bin 函数得到的 RSA 结构中的 n

返回值:

见后面的返回值错误码列表

1.32 使用 Openssl 生成一个私钥文件

```
int WINAPI DICNet_GenerateRsaPrivateKeyFile(int hic, WORD wPrivateKeyFileID, WORD  
wPrivateKeySize, const BYTE *cpbyPrivateKeyDataD, const BYTE *cpbyPrivateKeyDataN);
```

功能说明:

使用 Openssl 生成一个私钥文件

参数说明:

wPrivateKeyFileID	[in]私钥文件 ID
wPrivateKeySize	[in]私钥大小
cpbyPrivateKeyDataD	[in]私钥数据 D, 一般为用 BN_bn2bin 函数得到的 RSA 结构中的 d
cpbyPrivateKeyDataN	[in]私钥数据 N, 一般为用 BN_bn2bin 函数得到的 RSA 结构中的 n

返回值:

见后面的返回值错误码列表

1.33 锁定网络锁

```
int WINAPI DICNet_Lock(int hic);
```

功能说明:

锁定网络锁

参数说明:**返回值:**

见后面的返回值错误码列表

1.34 解锁网络锁

```
int WINAPI DICNet_UnLock(int hic);
```

功能说明：

解锁网络锁

参数说明：

返回值：

见后面的返回值错误码列表

1.35 获取当前路径

```
int WINAPI DICNet_GetPath(int hic, WORD awPath[256]);
```

功能说明：

获取当前路径

参数说明：

awPath [out]输出当前路径

返回值：

见后面的返回值错误码列表

二、Rockey6Smart 错误码定义

2.1 错误码常量列表

序号	错误码	值	描述
1	SCARD_S_SUCCESS	0x00000000L	成功，没有错误
2	SCARD_F_INTERNAL_ERROR	0x80100001L	内部连接检查失败
3	SCARD_E_CANCELLED	0x80100002L	操作被用户中止
4	SCARD_E_INVALID_HANDLE	0x80100003L	不正确的操作句柄
5	SCARD_E_INVALID_PARAMETER	0x80100004L	不正确的参数(p1, p2)
6	SCARD_E_INVALID_TARGET	0x80100005L	注册的启动信息丢失或无效
7	SCARD_E_NO_MEMORY	0x80100006L	没有足够的内存用于完成命令
8	SCARD_F_WAITED_TOO_LONG	0x80100007L	内部超时
9	SCARD_E_INSUFFICIENT_BUFFER	0x80100008L	用户给出的缓冲区太小，不足以放下全部的返回数据
10	SCARD_E_UNKNOWN_READER	0x80100009L	未知的读卡器
11	SCARD_E_TIMEOUT	0x8010000AL	用户指定的时间超时
12	SCARD_E_SHARING_VIOLATION	0x8010000BL	卡正在被其它连接占用
13	SCARD_E_NO_SMARTCARD	0x8010000CL	在读卡器里面没有卡
14	SCARD_E_UNKNOWN_CARD	0x8010000DL	未知的卡类型
15	SCARD_E_CANT_DISPOSE	0x8010000EL	读卡器无法完成退出卡操作
16	SCARD_E_PROTO_MISMATCH	0x8010000FL	当前的卡不支持用户指定的通讯协议
17	SCARD_E_NOT_READY	0x80100010L	卡还没有准备好接收命令
18	SCARD_E_INVALID_VALUE	0x80100011L	某些变量的值不合适
19	SCARD_E_SYSTEM_CANCELLED	0x80100012L	操作被系统中止，可能是重新登陆或关机
20	SCARD_F_COMM_ERROR	0x80100013L	内部通讯错误
21	SCARD_F_UNKNOWN_ERROR	0x80100014L	内部未知错误
22	SCARD_E_INVALID_ATR	0x80100015L	无效的ATR 串
23	SCARD_E_NOT_TRANSACTED	0x80100016L	用户尝试结束某个不存在的处理
24	SCARD_E_READER_UNAVAILABLE	0x80100017L	指定的读卡器当前无法使用
25	SCARD_P_SHUTDOWN	0x80100018L	操作被中止，允许服务程序退出
26	SCARD_E_PCI_TOO_SMALL	0x80100019L	PCI 的接收缓冲区太小
27	SCARD_E_READER_UNSUPPORTED	0x8010001AL	读卡器的驱动无法支持当

			前的读卡器
28	SCARD_E_DUPLICATE_READER	0x8010001BL	读卡器的驱动程序无法建立唯一的名字，已经有相同名字的读卡器存在
29	SCARD_E_CARD_UNSUPPORTED	0x8010001CL	卡无法被当前的读卡器支持
30	SCARD_E_NO_SERVICE	0x8010001DL	智能卡服务没有开启
31	SCARD_E_SERVICE_STOPPED	0x8010001EL	智能卡服务已经被中止
32	SCARD_E_UNEXPECTED	0x8010001FL	某个意外的智能卡错误产生
33	SCARD_E_ICC_INSTALLATION	0x80100020L	无法获知智能卡的提供者信息
34	SCARD_E_ICC_CREATEORDER	0x80100021L	无法获知智能卡的生产者信息
35	SCARD_E_UNSUPPORTED_FEATURE	0x80100022L	当前的智能卡无法支持用户要求的功能
36	SCARD_E_DIR_NOT_FOUND	0x80100023L	指定的目录不存在
37	SCARD_E_FILE_NOT_FOUND	0x80100024L	指定的文件不存在
38	SCARD_E_NO_DIR	0x80100025L	指定的目录不再是有效的目录
39	SCARD_E_NO_FILE	0x80100026L	指定的文件不再是有效的文件，没有选择文件
40	SCARD_E_NO_ACCESS	0x80100027L	此文件拒绝访问
41	SCARD_E_WRITE_TOO_MANY	0x80100028L	卡的空间已满，无法再写入信息
42	SCARD_E_BAD_SEEK	0x80100029L	设置文件指针错误
43	SCARD_E_INVALID_CHV	0x8010002AL	PIN 码错误
44	SCARD_E_UNKNOWN_RES_MNG	0x8010002BL	一个无法识别的错误码从智能卡服务返回
45	SCARD_E_NO_SUCH_CERTIFICATE	0x8010002CL	请求的证书不存在
46	SCARD_E_CERTIFICATE_UNAVAILABLE	0x8010002DL	请求的证书不允许获得
47	SCARD_E_NO_READERS_AVAILABLE	0x8010002EL	找不到任何一个读卡器
48	SCARD_E_COMM_DATA_LOST	0x8010002FL	智能卡通讯过程中发生数据丢失，请再次尝试
49	SCARD_E_NO_KEY_CONTAINER	0x80100030L	请求的密钥文件不存在
50	SCARD_W_UNSUPPORTED_CARD	0x80100065L	由于ATR 配置冲突，读卡器无法跟卡通讯
51	SCARD_W_UNRESPONSIVE_CARD	0x80100066L	卡对复位没有响应
52	SCARD_W_UNPOWERED_CARD	0x80100067L	卡没有电
53	SCARD_W_RESET_CARD	0x80100068L	卡被复位了，因此共享的信息无效了

54	SCARD_W_REMOVED_CARD	0x80100069L	卡已经被移出了
55	SCARD_W_SECURITY_VIOLATION	0x8010006AL	因为安全规则，访问被拒绝了
56	SCARD_W_WRONG_CHV	0x8010006BL	PIN 码没有被验证，访问被拒绝
57	SCARD_W_CHV_BLOCKED	0x8010006CL	已经到达最大PIN 码验证次数，访问被拒绝
58	SCARD_W_EOF	0x8010006DL	已经到达最后的智能卡文件，没有更多的文件可以访问了
59	SCARD_W_CANCELLED_BY_USER	0x8010006EL	操作被用户中止
60	SCARD_W_CARD_NOT_AUTHENTICATED	0x8010006FL	智能卡PIN 没有设置
61	SCARD_E_FILE_EXISTS	0xA0100001L	文件已经存在
62	SCARD_E_EPROM_ERROR	0xA0100002L	卡内存储器操作出错
63	SCARD_E_INVALID_CLA	0xA0100003L	用户给出了无效的CLA
64	SCARD_E_INVALID_INS	0xA0100004L	用户给出了无效的INS
65	SCARD_E_VM_ADDRESS_ERROR	0xA0100005L	VM 地址超界/异常
66	SCARD_E_ZERO_DIVIDE	0xA0100006L	除0 错
67	SCARD_E_WRONG_POSITION	0xA0100007L	卡没有被插入到正确的位置
68	SCARD_E_UNKNOWN_STATE	0xA0100008L	卡当前处于某种未知的状态
69	SCARD_E_CARD_NOT_OPENED	0xA0100009L	卡还没有被打开
70	SCARD_E_UNKNOWN_COMMAND	0xA010000AL	未知的命令
71	SCARD_E_ZERO_TRYTIME	0xA010000BL	即将设定超级密码的重新设置次数是0
72	SCARD_E_TOO_MANY_DEVICE	0xA010000CL	打开了太多的设备

2.2 虚拟设备专用错误码

序号	错误码	值	描述
1	SCARD_E_FILE_CREATE_FAILED	0xA0101001L	创建虚拟卡文件失败
2	SCARD_E_FILE_OPEN_FAILED	0xA0101002L	打开虚拟卡文件失败

2.3 网络锁专用代码

0xA0100101	Dic32u.dll发生异常
0xA0100102	网络错误
0xA0100103	没有找到Dic32u.dll
0xA0100104	参数错误

0xA0100105	模块定义文件没有找到
0xA0100109	服务端CacheSize出现错误。
0xA010010A	服务端的ROCKEY6设备损坏或被拔走
0xA010010B	服务端不支持此API
0xA010010C	相同的模块号在相同的进程中被打开
0xA010010D	网络传输错误，用NrGetLastError
0xA010010E	无效的句柄，可能是此句柄已经关闭
0xA010010F	网络锁硬件损坏
0xA0100110	客户端Dll被修改，拒绝提供服务
0xA0100111	服务端程序被修改，不再提供服务
0xA0100112	用户当前路径被其他用户删除，
0xA0100113	网络上找不到锁
0xA0100114	网络上找不到更多的锁
0xA0100115	已经被其他句柄锁定。
0xA0100116	此命令需要先锁定
0xA0100117	登录时数据校验失败
0xA0102001	输入的TOKENINFO校验出错
0xA0102002	服务变动，输入的ROOTKEY与卡内不一致
0xA0102003	同一个TOKENINFO重复登录
0xA0102004	模块号不匹配
0xA0102005	模块号超出合法范围
0xA0102006	指定模块数据找不到
0xA0102007	指定模块数据已经删除
0xA0102008	卡内模块数据校验值错
0xA0102009	已达最大用户数
0xA010200A	超过允许登录期限
0xA010200B	允许登录次数已减为0
0xA010200C	USERINFO校验值不正确

2.4 网络错误码

宏	代码	含义
ERR_INIT_SOCK	2001	初始化Winsock出错
ERR_NOSUCHPROTO	2002	没有与制定协议对应的服务
ERR_UDPSOCKCREATE	2003	UDP创建套接字失败
ERR_UDPSETBROADCAST	2004	UDP设置广播失败
ERR_UDPBINDFAILED	2005	UDP绑定失败
ERR_SVRCALLBACKNULL	2006	服务端回调函数为空
ERR_TCPSOCKCREATE	2007	TCP创建套接字失败
ERR_TCPBINDFAILED	2008	TCP绑定失败
ERR_TCPLISTENFAILED	2009	TCP侦听失败

ERR_NOSUCHSEACH	2010	不支持的搜索方式
ERR_UDPSSEND	2012	UDP发送数据失败
ERR_UDPTIMEOUT	2013	UDP等待数据超时
ERR_UDPrecv	2014	UDP接收数据失败
ERR_TCPCONNECT	2015	TCP连接服务器失败
ERR_TCPSENDTIMEOUT	2016	TCP等待发送超时
ERR_TCPSEND	2017	TCP发送数据失败
ERR_TCPRECVTIMEOUT	2018	TCP等待接收超时
ERR_TCPRECV	2019	TCP接受数据失败
ERR_IPXSOCKCREATE	2020	IPX创建套接字失败
ERR_IPXSETBROADCAST	2021	IPX设置广播失败
ERR_IPXSEND	2022	IPX发送数据失败
ERR_IPXRECV	2023	IPX接
ERR_IPXBIND	2024	IPX绑
ERR_NBSRESET	2025	NBS初
ERR_NBSADDNAME	2026	NBS加
ERR_NBSSEND	2027	NBS发
ERR_NBSRECV	2028	NBS接

2.5 扩展错误码

当API返回网络错误时就可以用NrGetLastError函数得到详细的错误原因。从网络错误码可以断定是哪种协议出现了错误，下面将NrGetLastError可能的返回值列出来供参考：
UDP/TCP和IPX的扩展错误码参考

宏	代码
WSABASEERR	10000
WSAEINTR	(WSABASEERR+4)
WSAEBADF	(WSABASEERR+9)
WSAEACCES	(WSABASEERR+13)
WSAEFAULT	(WSABASEERR+14)
WSAEINVAL	(WSABASEERR+22)
WSAEMFILE	(WSABASEERR+24)
WSAEWOULDBLOCK	(WSABASEERR+35)
WSAEINPROGRESS	(WSABASEERR+36)
WSAEALREADY	(WSABASEERR+37)
WSAENOTSOCK	(WSABASEERR+38)
WSAESTADDRREQ	(WSABASEERR+39)
WSAEMSGSIZE	(WSABASEERR+40)
WSAEPROTOPTYPE	(WSABASEERR+41)
WSAENOPROTOPT	(WSABASEERR+42)
WSAEPROTONOSUPPORT	(WSABASEERR+43)

WSAESOCKTNOSUPPORT	(WSABASEERR+44)
WSAEOPNOTSUPP	(WSABASEERR+45)
WSAEPFNOSUPPORT	(WSABASEERR+46)
WSAEAFNOSUPPORT	(WSABASEERR+47)
WSAEADDRINUSE	(WSABASEERR+48)
WSAEADDRNOTAVAIL	(WSABASEERR+49)
WSAENETDOWN	(WSABASEERR+50)
WSAENETUNREACH	(WSABASEERR+51)
WSAENETRESET	(WSABASEERR+52)
WSAECONNABORTED	(WSABASEERR+53)
WSAECONNRESET	(WSABASEERR+54)
WSAENOBUFS	(WSABASEERR+55)
WSAEISCONN	(WSABASEERR+56)
WSAENOTCONN	(WSABASEERR+57)
WSAESHUTDOWN	(WSABASEERR+58)
WSAETOOMANYREFS	(WSABASEERR+59)
WSAETIMEDOUT	(WSABASEERR+60)
WSAECONNREFUSED	(WSABASEERR+61)
WSAELoop	(WSABASEERR+62)
WSAENAMETOOLONG	(WSABASEERR+63)
WSAEHOSTDOWN	(WSABASEERR+64)
WSAEHOSTUNREACH	(WSABASEERR+65)
WSAENOTEMPTY	(WSABASEERR+66)
WSAEPROCLIM	(WSABASEERR+67)
WSAEUSERS	(WSABASEERR+68)
WSAEDQUOT	(WSABASEERR+69)
WSAESTALE	(WSABASEERR+70)
WSAEREMOTE	(WSABASEERR+71)
WSASYSNOTREADY	(WSABASEERR+91)
WSAVERNOTSUPPORTED	(WSABASEERR+92)
WSANOTINITIALISED	(WSABASEERR+93)
WSAEDISCON	(WSABASEERR+101)
WSAHOST_NOT_FOUND	(WSABASEERR+1001)
WSATRY_AGAIN	(WSABASEERR+1002)
WSANO_RECOVERY	(WSABASEERR+1003)
WSANO_DATA	(WSABASEERR+1004)

2.6 NetBios 扩展错误码参考

代码	含义
----	----

0x00	good return, also returned when ASYNCH request accepted
0x01	illegal buffer length
0x03	illegal command
0x05	command time out
0x06	message incomplete, issue another command
0x07	illegal buffer address
0x08	session number out of range
0x09	No resource available
0x0a	session closed
0x0b	command cancelled
0x0d	duplicate name
0x0e	name table full
0x0f	No deletions, name has active sessions
0x11	local session table full
0x12	remote session table full
0x13	illegal name number
0x14	no call name
0x15	cannot put * in NCB_NAME
0x16	name in use on remote adapter
0x17	name deleted
0x18	session ended abnormally
0x19	name conflict detected
0x21	interface busy, IRET before retrying
0x22	too many commands outstanding, retry later
0x23	ncb_lana_num field invalid
0x24	command completed while cancel occurring
0x26	command not valid to cancel
0x30	name defined by another local process
0x34	environment undefined. RESET required
0x35	required OS resources exhausted
0x36	max number of applications exceeded
0x37	no saps available for netbios
0x38	requested resources are not available
0x39	invalid ncb address or length > segment
0x3B	invalid NCB DDID
0x3C	lock of user area failed
0x3f	NETBIOS not loaded
0x40	system error
0xff	asynchronous command is not yet finished

三、附录

3.1 有关文件系统的说明

1 目录/文件ID (2 字节)

在加密锁内部，每个文件与目录都有一个ID，这是一个16 位的数字。在同一目录下的目录与文件的ID 必须不同，作为文件和目录的唯一标识。

备注:

3F00 是根目录ID

2F01 是ATR 文件的ID

3FFF 是当前目录的ID(我们没有使用)

0000 系统保留

在创建文件和目录的时候，不能够选择以上的ID

2 目录/文件类别编码(1 字节)

每个文件和目录都拥有一个自身的类别描述字节，我们称其为类别码(FCLA)，表示一个文件所属的类别，类别编码相同的文件一般会被假定为同一产品的文件。

加密锁内部的可执行文件受到类别码(FCLA)的限制，只能够读写和创建相同类别码(FCLA)的数据文件。

每个文件的类别码(FCLA)在IC 卡中占用1 字节，表示范围是0 - FF。

系统有一个字节用来记录当前的文件类别，我们称其为"系统当前文件类别"(SYSFCLA)。

作为特例FF(16 进制)表示"无类别"，FF 类别的数据文件可以被任何类别的可执行文件访问。

"系统当前文件类别" 在执行"文件选择" 命令的时候或调用加密锁中的可执行文件的时候，会产生切换SYSFCLA = FCLA。

如果切换的目标文件是"无类别" 文件，SYSFCLA 会保持原有类别不变。

当产生类别切换的时候，"系统当前文件权限" 会被重新置0 (SYSPRI = 0)

加密锁复位后SYSFCLA 缺省是FF (无类别)。

系统根目录的类别是"无类别"。

在加密锁中，相同类别的文件构成一个相对独立的子系统，"无类别" 数据文件是公共数据，"无类别" 可执行文件是公共处理程序。

无类别的可执行文件只能访问无类别的数据文件。

类别编码更为重要的用途是在可执行的过滤器文件上，以后我们会详细说明。

对使用者的建议:

通常类别编码被应用在开发者的管理上，如果某个开发商开发了多个产品，或多个开发商合作开发同一个产品，类别管理是很方便的做法。

如果某个加密锁只需要保护一套软件，建议类别码都设定为FF。

在同一目录下的文件和子目录建议设定相同的类别，用"文件权限"(即文件安全级别) 来控制访问。

3 目录/文件权限（即文件安全级别）编码(半字节)

除了类别以外，每个文件还有一个“文件的访问权限”(FPRI) 来限制相同类别文件的访问控制。

文件的访问权限(FPRI) 是针对相同类别文件设置的，也就是说FPRI是比FCLA 低一层的访问控制。

文件的访问权限在IC 卡中占用半个字节，编码范围是0-15，分为16 个不同的权限。0 是最低级，15是最高级。

系统有一个字节用来记录当前的文件的访问权限“系统当前文件访问权限”(SYSFPRI)

“系统当前文件访问权限”(SYSFPRI) 必须由加密锁内部的可执行文件来提升，SYSFPRI = 可执行文件通过系统调用设定的权限，但可执行文件不能设定超过自身设定的权限
SYSFPRI <= 可执行文件的FPRI。

可执行文件只能操作低于或等于自身级别的数据文件(读、写、创建)。

可执行文件在退出的时候可以清除“系统当前文件访问权限”(SYSFPRI = 0)

对加密锁内部的可执行文件而言，可以设定一个文件属性FILEATTR_UPIGNORE，这个属性告诉系统，如果SYSFPRI >= 可执行文件的FPRI，那么就不需要调用这个可执行文件了。

对使用者的建议：

权限编码是更高层次的文件管理功能，如果用户不需要这么复杂的管理功能，把所有的文件权限编码都设定为0 就好了。

注意权限编码跟类别编码不同，先分类再分级。//

4 目录/文件属性(半字节)

每个文件/目录都有一个属性描述(FATTR)，每位代表一种属性。

FILEATTR_NORMAL	0x00	// 普通文件	a)
FILEATTR_EXEC	0x10	// 可执行文件	b)
FILEATTR_DIR	0x20	// 这是一个目录，不是文件	c)
FILEATTR_UPIGNORE	0x40	// 高于忽略	d)
FILEATTR_INTERNAL	0x80	// 内部文件	e)

a) 普通文件

b) 可执行属性，表明这个文件是加密锁内部可执行文件，这种文件必须被放在加密锁的根目录下，而且只有在超级密码验证通过后才能创建和删除。

c) 目录属性，表示这是一个目录，如果这个属性被设置，其它属性位会被忽略。

d) 高于忽略属性，只对加密锁内部的可执行文件有效的属性位，如果SYSFPRI >= FPRI (即系统当前文件安全级别 >= 这个可执行文件的安全级别)，这个执行被忽略。

e) 内部文件属性位，表示这个文件只能由可执行文件来访问，外部操作只有在超级密码验证通过后可以删除，但不可以读写。在没有超级密码验证通过的情况下绝对不允许外部访问。

5 文件的长文件名

在创建文件/目录的时候，可以给出一个不超过16 个字符的长文件名，这个长文件名是可选的。如果设定了长文件名，以后对这个文件的操作既可以通过文件ID 也可以通过长文件名来访问。

3.2 其它功能的说明

创建目录的时候要给出目录ID、类别、权限、长文件名信息。

列目录指令采用的是一种枚举的方式，一直到最后返回错误，表示当前目录下没有更多的文件或目录了。

DIC_SetCurrentDir 类似于DOS 命令cd xxxx，这个命令只需要给出目录ID 或长文件名就可以，如果通过长文件名来设定，必须首先把ID 添为0。

创建文件的时候也要给出全部的文件ID、类别、权限、属性、文件大小、长文件名等信息，如果没有长文件名，全部添0 就可以。

创建文件会自动把被创建的文件设定为当前文件，不需要调用DIC_SetCurrentFile