



# 网络模块开发指南

V1.0

北京坚石诚信科技有限公司

网址: <http://www.jansh.com.cn>

修订记录:

修订日期	版本	修订内容
2013 年 7 月	V1.0	第一版发布

## 软件开发协议

坚石诚信科技股份有限公司（以下简称坚石）的所有产品，包括但不限于：开发工具包，磁盘，光盘，硬件设备和文档，以及未来的所有定单都受本协议的制约。如果您不愿接受这些条款，请在收到后的 7 天内将开发工具包寄回坚石，预付邮资和保险。我们会把货款退还给您，但要扣除运费和适当的手续费。

### 1. 许可使用

您可以将本软件合并、连接到您的计算机程序中，但其目的只是如开发指南中描述的那样保护该程序。您可以以存档为目的复制合理数量的拷贝。

### 2. 禁止使用

除在条款 1 中特别允许的之外，不得复制、反向工程、反汇编、反编译、修改、增加、改进软件、硬件和产品的其它部分。禁止对软件和产品任何部分进行反向工程，或企图推导软件的源代码。禁止使用产品中的磁性或光学介质来传递、存储非本产品的原始程序或由坚石提供的产品升级的任何数据。禁止将软件放在服务器上传播。

### 3. 有限担保

坚石保证在自产品交给您之日起的 12 个月内，在正常的使用情况下，硬件和软件存储介质没有重大的工艺和材料上的缺陷。

### 4. 修理限度

当根据本协议提出索赔时，坚石唯一的责任就是根据坚石的选择，免费进行替换或维修。坚石对更换后的任何产品部件都享有所有权。

保修索赔单必须在担保期内写好，在发生故障 14 天内连同令人信服的证据交给坚石。当将产品返还给坚石或坚石的授权代理商时，须预付运费和保险。

除了在本协议中保证的担保之外，坚石不再提供特别的或隐含的担保，也不再对本协议中所描述的产品负责，包括它们的质量，性能和对某一特定目的适应性。

### 5. 责任限度

不管因为什么原因，不管是因合同中的规定还是由于刑事的原因，包括疏忽的原因，而使您及任何一方受到了损失，由我方产品所造成的损失或该产品是起诉的原因或与起诉有间接关系，坚石对您及任何一方所承担的全部责任不超出您购买该产品所支付的货款。在任何情况下，坚石对于由于您不履行责任所导致的损失，或对于数据、利润、储蓄或其它的后续的和偶然的损失，即使坚石被建议有这种损失的可能性，或您根据第 3 方的索赔而提出的任何索赔均不负责任。

### 6. 协议终止

当您不能遵守本协议所规定的条款时，将终止您的许可和本协议。但条款 2， 3， 4， 5 将继续有效。

## Software Developer's Agreement

All Products of Jansh Technologies Co., Ltd. (Jansh) 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 Jansh provided enhancement or upgrade to the Product.
3. Warranty – Jansh 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, Jansh's sole obligation is to replace or repair, at the discretion of Jansh, any Product free of charge. Any replaced Product becomes the property of Jansh.

Warranty claims must be made in writing to Jansh 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 Jansh. Any Products that you return to Jansh, or a Jansh 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 Jansh's Liability – Jansh'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 Jansh 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 Jansh 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.

## 目录

ET-ARM 网络模块说明.....	2
ET-ARM 网络模块 API 接口.....	3
1.Dongle_Enum 枚举加密锁.....	3
2.Dongle_Open 打开加密锁 .....	4
3.Dongle_Close 关闭加密锁 .....	4
4.Dongle_GenRandom 产生随机数.....	5
5.Dongle_LEDControl LED 控制 .....	5
6.Dongle_ReadFile 读取数据文件 .....	6
7.Dongle_VerifyPin 校验密码 .....	6
8.Dongle_ReadData 读取数据存储区 .....	7
9.Dongle_WriteData 写入数据存储区 .....	7
10.Dongle_RsaPri RSA 私钥运算.....	8
11.Dongle_RsaPub RSA 公钥运算.....	9
12.Dongle_EccSign ECC 私钥签名 .....	10
13.Dongle_EccVerify ECC 公钥验签 .....	10
14.Dongle_TDES 3DES 运算.....	11
15.Dongle_Seed 种子码运算.....	12
16.Dongle_GetUTCtime 获取加密锁的 UTC 时间 .....	12
17.Dongle_GetDeadline 获取加密锁到期时间 .....	12
18.Dongle_RunExeFile 运行可执行程序 .....	13
错误码.....	14

## ET-ARM 网络模块说明

ET-ARM 网络模块是一个单独的服务程序，增加了网络功能，从原有的本地找锁升级到了网络找锁，从而实现了软件使用数量的限制以及授权管理的目的。ET-ARM 系列加密锁中任何型号的产品都可以使用网络模块，但是需要对加密硬件本身进行初始化操作，我们将使用了网络模块初始化工具初始化后的加密锁称之为网络锁。从硬件上看，网络锁与单机锁没有任何区别，但是单机锁与网络锁使用不同的 API 和服务程序，简单的理解就是，同一把 ET-ARM 加密锁，如果使用的是单机版加密锁的 SDK，那么它就是一把单机锁，如果使用的是网络版加密锁的 SDK，并且配合服务器程序一起使用，那么它就是一把网络锁。

从技术实现的角度讲，ET-ARM 网络锁仅仅实现了 ET-ARM 单机锁部分接口，例如，ET-ARM 单机锁可以调用 API 接口来实现下载可执行程序，但是 ET-ARM 网络锁就没有提供这样的接口，不过并不是说 ET-ARM 网络锁不能实现这样的功能，可以通过调用单机锁的 API 接口来实现，或者直接使用 ET-ARM 加密锁集成编辑工具（RyARMTTool）来进行操作。因此，ET-ARM 网络锁其实也完全可以当做 ET-ARM 单机锁来使用，但是 ET-ARM 单机锁如果没有经过网络锁初始化工具（NetLockInitTool）进行初始化，那么是不能实现 ET-ARM 网络锁功能的。有关 ET-ARM 单机锁的工具使用及功能介绍请查看《ET-ARM 用户工具使用手册》，在了解了 ET-ARM 单机锁的功能后，对如何更好的使用 ET-ARM 网络锁会有很大的帮助。

ET-ARM 网络锁可以允许多个软件的实例同时访问连接在服务器上的同一个加密锁，即使对加密锁进行了共享。通过限制同时运行的客户端软件数目，可以达到对软件的使用进行控制的目的。相对于 ET-ARM 单机锁来说，ET-ARM 网络锁不仅需要保护软件不被非法复制，同时还能够严格的限制同时运行的软件数目。

ET-ARM 网络锁是将 ET-ARM 单机锁的功能实现在了网络上，但是这完全不会影响到它对软件的保护强度，而是增加了 ET-ARM 系列加密锁的功能，扩展了 ET-ARM 系列加密锁的应用范围。

本手册将介绍 ET-ARM 网络模块所提供的 API 以及相关错误码，通过参考本手册，可以帮助开发商灵活的运用 ET-ARM 网络模块。

# ET-ARM 网络模块 API 接口

## 1.Dongle\_Enum 枚举加密锁

*DWORD WINAPI Dongle\_Enum(DONGLE\_INFO \* pDongleInfo, int \* pCount);*

说明:

本函数最多会枚举出 32 个 HID 设备和 32 个 CCID 设备。

参数:

pDongleInfo            [out]        设备信息的数组。当此参数为 NULL 时, pCount 返回找到的设备的数目。

pCount                [out]        设备数目。该函数最多可以同时枚举出 32 个 HID 设备和 32 个 CCID 设备。

返回值:

DONGLE\_SUCCESS            执行成功。

注意:

为能使后续的介绍更加方便, 先将加密锁信息相关的结构体和定义进行说明:

```
#define CONST_PID                            0xFFFFFFFF    //出厂时默认 PID
#define CONST_USERPIN                        "12345678"    //出厂时默认 USERPIN
#define CONST_ADMINPIN                       "FFFFFFFFFFFFFF" //出厂时默认 ADMINPIN
```

默认的 PIN 码重试次数为无限制。根据种子码初始化锁时会同时初始化 PID 和 ADMINPIN (PID 不可更改, ADMINPIN 可更改)。

锁的信息:

```
typedef struct
{
    unsigned short m_Ver;                    //COS 版本, 比如: 0x0201, 表示 2.01 版
    unsigned short m_Type;                   //产品类型: 0xFF 表示标准版, 0x00 为时钟锁, 0x01 为带时钟的 U 盘锁, 0x02 为标准 U 盘锁
    unsigned char m_BirthDay[8];            //出厂日期
    unsigned long m_Agent;                   //代理商编号, 比如: 默认的 0xFFFFFFFF
    unsigned long m_PID;                    //产品 ID
    unsigned long m_UserID;                  //用户 ID
    unsigned char m_HID[8];                  //8 字节的硬件 ID
```



```
unsigned long    m_IsMother;           //母锁标志：0x01 表示是母锁，0x00 表示不是母锁
unsigned long    m_DevType;           //设备类型(PROTOCOL_HID 或者 PROTOCOL_CCID)
} DONGLE_INFO;
```

## 2.Dongle\_Open 打开加密锁

*DWORD WINAPI Dongle\_Open(DONGLE\_HANDLE \* phDongle, int nIndex);*

说明：

打开指定的加密锁的指定模块号。

参数：

phDongle            [out]        句柄指针。如果打开成功,会被填充。

nIndex             [in]        基于 0 的索引值。指示打开找到的第几把加密锁。

返回值：

DONGLE\_SUCCESS        打开成功。

注意：

对索引号的定义：

#define

MAKEDWORD(x,y)((DWORD)((((DWORD)(x)<<16)&0xFFFF0000)|((DWORD)((((DWORD)(y)&0x0000FFFF))))

其中 x 表示从 0 开始的加密锁的索引值，y 表示准备打开的模块号（取值范围：0~63）。例如打开第 0 把锁的第 2 个模块，参数 nIndex 的值为 MAKEDWORD(0,1)。

## 3.Dongle\_Close 关闭加密锁

*DWORD WINAPI Dongle\_Close(DONGLE\_HANDLE hDongle);*

说明：

关闭打开的加密锁。

参数：

hDongle            [in]        打开的加密锁句柄。

返回值：

DONGLE\_SUCCESS        关闭成功。

## 4.Dongle\_GenRandom 产生随机数

*DWORD WINAPI Dongle\_GenRandom(DONGLE\_HANDLE hDongle, int nLen, BYTE \* pRandom);*

说明:

匿名权限即可操作。

参数:

hDongle	[in]	打开的加密锁句柄。
nLen	[in]	要产生的随机数的长度，len 的取值范围为 1~128。
pRandom	[out]	随机数缓冲区。

返回值:

DONGLE\_SUCCESS                  获取随机数成功。

## 5.Dongle\_LEDControl LED 控制

*DWORD WINAPI Dongle\_LEDControl(DONGLE\_HANDLE hDongle, int nFlag);*

说明:

LED 灯的控制操作。匿名权限即可操作。

参数:

hDongle	[in]	打开的加密锁句柄。
nFlag	[in]	控制类型，例如：flag = LED_ON，表示控制 LED 为亮的状态； flag = LED_OFF，表示控制 LED 为灭的状态； flag = LED_BLINK，表示控制 LED 为闪烁的状态。

返回值:

DONGLE\_SUCCESS                  命令执行成功。

注意:

LED 灯状态定义如下:

```
#define LED_OFF                      0 //灯灭
#define LED_ON                        1 //灯亮
```

```
#define LED_BLINK
```

```
2 //灯闪
```

## 6.Dongle\_ReadFile 读取数据文件

```
DWORD WINAPI Dongle_ReadFile(DONGLE_HANDLE hDongle, WORD wFileID, WORD wOffset, BYTE* pOutData, int nDataLen);
```

说明：

读取加密锁内的数据文件。数据文件的读取权限取决于创建时的设定。

参数：

hDongle	[in]	打开的加密锁句柄。
wFileID	[in]	文件 ID。
wOffset	[in]	文件偏移量。
pOutData	[in]	数据缓冲区。
nDataLen	[out]	参数 pOutData 的长度，读取的最大长度不能超过 1024 个字节

返回值：

DONGLE\_SUCCESS          读取数据文件成功

## 7.Dongle\_VerifyPin 校验密码

```
DWORD WINAPI Dongle_VerifyPin(DONGLE_HANDLE hDongle, int nFlags, char* pPIN, int* pRemainCount);
```

说明：

校验密码。

参数：

hDongle	[in]	打开的加密锁句柄。
nFlags	[in]	PIN 码类型。默认参数取值为 FLAG_USERPIN。
pPIN	[in]	PIN 码。
pRemainCount	[out]	剩余重试次数。返回 0x9000 表示已锁死；1~253 表示剩余次数；255 表示不限制重试次数。

返回值：

DONGLE\_SUCCESS

校验成功。

如果校验失败，函数的返回值中也含有剩余的重试次数，（错误码 & 0xFFFFF00）== DONGLE\_INCORRECT\_PIN，即后两位表示剩余次数。

注意：

PIN 码类型定义如下：

```
#define FLAG_USERPIN          0 //用户 PIN
#define FLAG_ADMINPIN        1 //开发商 PIN
```

## 8.Dongle\_ReadData 读取数据存储区

*DWORD WINAPI Dongle\_ReadData(DONGLE\_HANDLE hDongle, int nOffset, BYTE\* pData, int nDataLen);*

说明：

读取锁内数据区数据。数据区大小共 8k，前 4k(0~4095)的读写没有权限限制，后 4k(4096~8191)任意权限可读，但是不可写。

参数：

hDongle	[in]	打开的加密锁句柄。
nOffset	[in]	起始偏移。范围在 0~8191。
pData	[out]	读取的数据缓冲区。
nDataLen	[in]	参数 pData 的缓冲区大小，最大不能超过 1024 个字节

返回值：

DONGLE\_SUCCESS 读取数据区数据成功。

## 9.Dongle\_WriteData 写入数据存储区

*DWORD WINAPI Dongle\_WriteData(DONGLE\_HANDLE hDongle, int nOffset, BYTE\* pData, int nDataLen);*

说明：

写入锁内数据区数据。数据区大小共 8k，前 4k(0~4095)的读写没有权限限制，后 4k(4096~8191)任意权限可读，但是不可写。

参数：

hDongle	[in]	打开的加密锁句柄。
nOffset	[in]	起始偏移。范围在 0~8191。
pData	[in]	写入的数据缓冲区。
nDataLen	[in]	参数 pData 的缓冲区大小，最大不能超过 1024 个字节
返回值：		
DONGLE_SUCCESS		写入数据区数据成功。

## 10.Dongle\_RsaPri RSA 私钥运算

*DWORD WINAPI Dongle\_RsaPri(DONGLE\_HANDLE hDongle, WORD wPriFileID, int nFlag, BYTE\* pInData, int nInDataLen, BYTE\* pOutData, int\* pOutDataLen);*

说明：

函数的使用权限取决于锁内 RSA 私钥文件的权限，在 RSA 私钥文件创建时设定。

参数：

hDongle	[in]	打开的加密锁句柄。
wPriFileID	[in]	RSA 私钥文件 ID。
nFlag	[in]	运算类型。例如，FLAG_ENCODE 表示加密运算；FLAG_DECODE 表示解密运算。
pInData	[in]	输入数据。
nInDataLen	[in]	参数 pInData 的大小
pOutData	[out]	输出数据缓冲区。
pOutDataLen	[in,out]	参数 pOutData 的大小和返回的数据大小。
返回值：		
DONGLE_SUCCESS		RSA 私钥运算成功。

注意：

- 1.对于加密运算,输入数据长度必须小于私钥 ID 为 wPriFileID 的密钥位数除以 8 减去 11，以便在函数内部进行 padding
- 2.对于解密运算,输入数据长度必须与 wPriFileID 中指示的密钥位数除以 8 的值一致(比如 1024 位密钥时为 128，2048 时为 256)

3.加密时内部 padding 方式为:PKCS1\_TYPE\_1 (即第二个字节为 0x01,空数据填充 0xFF)

加解密标志定义如下:

```
#define FLAG_ENCODE          0 //加密
#define FLAG_DECODE         1 //解密
```

## 11.Dongle\_RsaPub RSA 公钥运算

***DWORD WINAPI Dongle\_RsaPub(DONGLE\_HANDLE hDongle, int nFlag, RSA\_PUBLIC\_KEY\* pPubKey, BYTE\* pInData, int nInDataLen, BYTE\* pOutData, int\* pOutDataLen);***

说明:

RSA 公钥运算, 匿名权限可调用。

参数:

hDongle	[in]	打开的加密锁句柄。
nFlag	[in]	运算类型。例如, FLAG_ENCODE 表示加密运算; FLAG_DECODE 表示解密运算。
pPubKey	[in]	RSA 公钥数据。该数据来源于生成 RSA 公私钥时的公钥数据。
pInData	[in]	输入数据。
nInDataLen	[in]	参数 pInData 的大小。
pOutData	[out]	输出数据缓冲区。
pOutDataLen	[in,out]	参数 pOutData 的大小和返回的数据大小。

返回值:

DONGLE\_SUCCESS                  RSA 公钥运算成功。

注意:

- 1.对于加密运算,输入数据长度必须小于 pPubKey 中指示的密钥位数除以 8 减去 11, 以便在函数内部进行 padding ;
- 2.对于解密运算,输入数据长度必须与 pPubKey 中指示的密钥位数除以 8 的值一致(比如 1024 位密钥时为 128, 2048 时为 256) ;
- 3.加密时内部 padding 方式为:PKCS1\_TYPE\_2 (即第二个字节为 0x02,空数据填充随机数) 。

## 12.Dongle\_EccSign ECC 私钥签名

**DWORD WINAPI Dongle\_EccSign(DONGLE\_HANDLE hDongle, WORD wPriFileID, BYTE\* pHashData, int nHashDataLen, BYTE\* pOutData);**

说明:

函数的使用权限取决于锁内 ECC 私钥文件的权限，在 ECC 私钥文件创建时设定。

参数:

hDongle	[in]	打开的加密锁句柄。
wPriFileID	[in]	ECC 私钥文件 ID。
pHashData	[in]	Hash 数据。
nHashDataLen	[in]	参数 pHash 的大小。
pOutData	[out]	签名数据。大小固定为 64 字节(256 位 ECC 时长度刚好,192 位 ECC 时多余的位会补 0)。

返回值:

DONGLE\_SUCCESS                  表示签名成功。

注意:

1.锁内签名算法为: ECDSA\_Sign

2.输入的 Hash 值的长度与 ECC 私钥的密钥长度有关,如果密钥是 192 位的,则 hash 值长度不能超过 24(192/8 = 24)字节, 如果密钥是 256 位的,则 hash 值长度不能超过 32(256/8 = 32)字节)

3.曲线参数为:EC\_NIST\_PRIME\_192 及 EC\_NIST\_PRIME\_256

## 13.Dongle\_EccVerify ECC 公钥验签

**DWORD WINAPI Dongle\_EccVerify(DONGLE\_HANDLE hDongle, ECCSM2\_PUBLIC\_KEY\* pPubKey, BYTE\* pHashData, int nHashDataLen, BYTE\* pSign);**

说明:

无

参数:

hDongle	[in]	打开的加密锁句柄。
---------	------	-----------

pPubKey	[in]	ECC 公钥数据。
pHashData	[in]	Hash 数据。
nHashDataLen	[in]	参数 pHashData 的大小。
pSign	[in]	签名数据。大小固定为 64 字节，为 Dongle_EccSign 函数返回的 pOut 数据。

返回值：

DONGLE\_SUCCESS                      表示验签成功,否则表示验签失败。

注意：

1.锁内签名算法为: ECDSA\_Verify

2.输入的 Hash 值的长度与 ECC 私钥的密钥长度有关,如果密钥是 192 位的,则 hash 值长度不能超过 24(192/8 = 24)字节, 如果密钥是 256 位的,则 hash 值长度不能超过 32(256/8 = 32)字节。

3.曲线参数为:EC\_NIST\_PRIME\_192 及 EC\_NIST\_PRIME\_256

## 14.Dongle\_TDES 3DES 运算

*DWORD WINAPI Dongle\_TDES(DONGLE\_HANDLE hDongle, WORD wKeyFileID, int nFlag, BYTE\* pInData, BYTE\* pOutData, int nDataLen);*

说明：

解密运算匿名权限即可，加密运算的权限取决于密钥文件的权限。

参数：

hDongle	[in]	打开的加密锁句柄。
wKeyFileID	[in]	密钥文件 ID。
nFlag	[in]	运算类型。例如，FLAG_ENCODE 表示加密运算；FLAG_DECODE 表示解密运算。
pInData	[in]	输入数据缓冲区。
pOutData	[out]	输出数据缓冲区。大小至少要和输入数据缓冲区相同，输入和输出数据缓冲区可以为同一个。
nDataLen	[in]	参数 pInData 的数据大小。数据长度必须是 16 的整数倍,允许的最大值是 1024。



## 15.Dongle\_Seed 种子码运算

*DWORD WINAPI Dongle\_Seed(DONGLE\_HANDLE hDongle, BYTE\* pSeed, int nSeedLen, BYTE\* pOutData);*

说明：

匿名权限可使用，若是时钟锁，时间到期后不能用。

参数：

hDongle	[in]	打开的加密锁句柄。
pSeed	[in]	输入的种子码数据。
nSeedLen	[in]	种子码长度。取值范围为 1~250 字节。
pOutData	[out]	输出数据缓冲区。输出的大小固定为 16 字节。

返回值：

DONGLE\_SUCCESS 种子码运算成功。

注意：

- 1.种子码算法与 PID 有关，空锁(即 PID=FFFFFFFF)不能进行种子码运算。
- 2.如果内部种子码可运算次数不为-1，当其递减到 0 后此函数将不能使用。

## 16.Dongle\_GetUTCTime 获取加密锁的 UTC 时间

*DWORD WINAPI Dongle\_GetUTCTime(DONGLE\_HANDLE hDongle, DWORD \* pdwUTCTime);*

参数：

hDongle	[in]	打开的加密锁句柄。
pdwUTCTime	[out]	时间值。

返回值：

DONGLE\_SUCCESS 获取 UTC 时间成功。

## 17.Dongle\_GetDeadline 获取加密锁到期时间

*DWORD WINAPI Dongle\_GetDeadline(DONGLE\_HANDLE hDongle, DWORD \* pdwTime);*

说明：

匿名权限即可获取。

参数：

**hDongle**            [in]            打开的加密锁句柄。

**pdwTime**           [out]            获取的到期 UTC 时间值。

若 \*pdwTime = 0xFFFFFFFF，表示不限制到期时间

若 (\*pdwTime & 0xFFFF0000) == 0，值表示还剩余几小时

若 (\*pdwTime & 0xFFFF0000) != 0，值表示到期的 UTC 的时间，可以通过 **gmtime** 等将该值进行转换。

返回值：

**DONGLE\_SUCCESS**            获取加密锁到期时间成功。

## 18.Dongle\_RunExeFile 运行可执行程序

***DWORD WINAPI Dongle\_RunExeFile(DONGLE\_HANDLE hDongle, WORD wFileID, BYTE \* pInOutBuf, WORD wInOutBufLen, int \* pMainRet);***

说明：

调用权限依据下载可执行文件时的设定。

参数：

**hDongle**            [in]            打开的加密锁句柄。

**wFileID**            [in]            可执行文件的文件 ID。

**pInOutBuf**           [in,out]            输入输出数据缓冲区。

**wInOutBufLen**       [in]            输入输出数据缓冲区 *pInOutBuf* 的大小。

**pMainRet**           [out]            锁内可执行程序 **main** 函数的返回值，可以为 **NULL**。

返回值：

**DONGLE\_SUCCESS**            运行可执行程序成功。

## 错误码

//错误码

#define	DONGLE_SUCCESS	0x00000000	操作成功
#define	DONGLE_NOT_FOUND	0xF0000001	未找到指定的设备
#define	DONGLE_INVALID_HANDLE	0xF0000002	无效的句柄
#define	DONGLE_INVALID_PARAMETER	0xF0000003	参数错误
#define	DONGLE_COMM_ERROR	0xF0000004	通讯错误
#define	DONGLE_INSUFFICIENT_BUFFER	0xF0000005	缓冲区空间不足
#define	DONGLE_NOT_INITIALIZED	0xF0000006	产品尚未初始化 (即没设置 PID)
#define	DONGLE_ALREADY_INITIALIZED	0xF0000007	产品已经初始化 (即已设置 PID)
#define	DONGLE_ADMINPIN_NOT_CHECK	0xF0000008	开发商密码没有验证
#define	DONGLE_USERPIN_NOT_CHECK	0xF0000009	用户密码没有验证
#define	DONGLE_INCORRECT_PIN	0xF000FF00	密码不正确 (后 2 位指示剩余次数)
#define	DONGLE_PIN_BLOCKED	0xF000000A	PIN 码已锁死
#define	DONGLE_ACCESS_DENIED	0xF000000B	访问被拒绝
#define	DONGLE_FILE_EXIST	0xF000000E	文件已存在
#define	DONGLE_FILE_NOT_FOUND	0xF000000F	未找到指定的文件
#define	DONGLE_READ_ERROR	0xF0000010	读取数据错误
#define	DONGLE_WRITE_ERROR	0xF0000011	写入数据错误
#define	DONGLE_FILE_READ_ERROR	0xF0000013	读取文件错误
#define	DONGLE_FILE_WRITE_ERROR	0xF0000014	写入文件错误
#define	DONGLE_FAILED	0xF0000016	操作失败
#define	DONGLE_CLOCK_EXPIRE	0xF0000017	加密锁时钟到期
#define	DONGLE_ERROR_UNKNOWN	0xFFFFFFFF	未知的错误

//网络锁相关操作

#define	DONGLE_FAILED_NET_CONNECTED	0xF0000018	已经建立连接
#define	DONGLE_FAILED_NET_FULL	0xF0000019	连接数已满
#define	DONGLE_FAILED_NET_NOTFINDNETLOCK	0xF0000020	没有找到可用的网络加密锁
#define	DONGLE_FAILED_NET_NOCONNECTED	0xF0000021	还没有建立连接
#define	DONGLE_FAILED_NET_DELETED	0xF0000022	客户端已经断开
#define	DONGLE_FAILED_NET_DENIED	0xF0000023	客户被拒绝访问服务器

//网络通讯返回值

#define	R_CONNECTFAILED	0xC0000001	连接失败
#define	R_SENDDATAFAILED	0xC0000002	发送数据失败
#define	R_SENDDATATIMEOUT	0xC0000003	发送数据超时
#define	R_CREATESOCKETFAILED	0xC0000004	创建套接字失败
#define	R_RECVDATATIMEOUT	0xC0000005	接收数据超时
#define	R_RECVDATAERROR	0xC0000006	接收数据错误