

ET99 多功能锁全功能 ActiveX 控件参考手册

概述

这一节概述利用 ET99 全功能 ActiveX 控件来编写 ET99 的程序。你可以在任何支持 COM 或 OLE 技术的开发环境中使用这个 ActiveX 控件，如下面的平台可以使用：

- Microsoft Visual C++
- Microsoft Visual Basic
- Inprise C++ Builder
- Inprise Delphi

你可以在 SDK 包的 samples 的子目录下看到相关的事例代码。

ET99 全功能 ActiveX 控件覆盖了所有的 API 提供的接口函数。它提供了如下的接口：

IET99MOD 接口覆盖所有的函数接口：

CLSID: {1AD79C4B-0D2A-4C67-BE0B-6DD608CE2BCC}

IID: { 1AD79C4B-0D2A-4C67-BE0B-6DD608CE2BCC}

ProgID: ET99_MOD. ET99MOD. 1

FindToken

函数 **FindToken** 查看计算机连接有多少个指定 pid 的 ET99。

函数原型：

HRESULT

```
FindToken( [in] BYTE* pid,  
           [out] int* count  
);
```

参数：

pid

[in] 这个参数指定要查找设备的 pid，为固定长度的 8 个字节长度的字符；

count

[out] 返回找到的设备的个数。

注解：

这个函数一般用在调用的第一个函数，用于查看计算机上连接了 ET99 的个数，然后再根据连接的个数来打开第几个设备。

返回值：

ET_SUCCESS: 执行成功,Count 为查找到的数目；

ET_UNIT_NOT_FOUND:没有可以用的硬件，此时 Count 值为 0。

OpenToken

函数 **OpenToken** 用来打开指定的 ET99 设备。

函数原型：

HRESULT

```
OpenToken(  
           [in] BYTE* pid,  
           [in] int index  
);
```

参数：

pid

[in] 这个参数指定要打开设备的 pid，为固定长度的 8 个字节长度的字符；

index

[in] 这个参数用来指定要打开的是第几个 ET99 的设备。

注解：

查看计算机上有多少个 ET99 以后，再利用该函数来打开指定的设备，以后的函数调用都必须基于这个函数调用。

返回值：

ET_SUCCESS: 执行成功。

ET_UNIT_NOT_FOUND: 打开指定的设备失败。

CloseToken

函数 **CloseToken** 用来关闭已经打开的 ET99 设备。

函数原型：

HRESULT

CloseToken();

要求：

OpenToken。

参数：

无

注解：

用来关闭已经打开的 ET99 设备。

返回值：

ET_SUCCESS: 执行成功。

ET_COMMUNICATIONS_ERROR: 没有打开设备。

VerifyPIN

函数 **VerifyPIN** 用来验证 PIN 码，以获得相应的安全状态。

函数原型：

HRESULT

VerifyPIN(

[in] int Flags,

[in] unsigned char* pucPIN

);

要求：

OpenToken。

参数：

IFlags

[in] 验证 Pin 码的类型，取值如下：

ET_VERIFY_USERPIN 验证的为普通用户 PIN 码

ET_VERIFY_SOPIN 验证的是超级用户 PIN 码

pucPIN

[in] 指向一个包含验证 pin 码的缓冲区，长度固定为 16 个字节长。

注解：

如果验证普通用户 PIN 码或者超级用户 PIN 错误，并且错误值在 0xF0 和 ET_PIN_ERR_MAX (0xFF) 之间,则返回值的后四位为剩余重试的次数。如果返回的是 0xFF 表示验证错误，且 PIN 码的重试次数为无限次。

返回值：

ET_SUCCESS：表示成功；

ET_INVALID_PARAMETER：无效的参数；

ET_COMMUNICATIONS_ERROR：没有打开设备。

ChangeUserPIN

函数 **ChangeUserPIN** 用来更改普通用户的 PIN 码。

函数原型：

HRESULT

ChangeUserPIN (

 [in] BYTE* pucOldPIN,

 [in] BYTE* pucNewPIN

);

要求：

OpenToken。

参数：

pucOldPIN

[in] 指向一个包含旧的普通用户 pin 码的缓冲区。PIN 码长度为固定 16 个字节。

pucNewPIN

[in] 指向一个包含新的普通用户 pin 码的缓冲区。PIN 码长度为固定 16 个字节。

注解：

该函数在更改普通用户 PIN 码前先验证原来的 PIN 码，如果验证普通用户 PIN 码错误，并且错误值在 0xF0 和 ET_PIN_ERR_MAX (0xFF) 之间,则返回值的后四位为剩余重试的次数。如果返回的是 0xFF 表示验证错误，且 PIN 码的重试次数为无限次。

返回值：

ET_SUCCESS：表示成功。

ET_HARD_ERROR: 硬件错误
ET_NOT_SET_PID: 没有设置 PID
ET_INVALID_PARAMETER: 无效的参数。
ET_COMMUNICATIONS_ERROR: 没有打开设备。

GenSoPIN

函数 **GenSoPIN** 根据指定的种子，产生超级用户 PIN 码。

函数原型:

HRESULT

```
GenSoPIN(  
    [in] Long SeedLen,  
    [in] BYTE* pucSeed,  
    [out] BYTE* pucNewSoPIN  
);
```

要求:

OpenToken。

参数:

SeedLen

[in] 种子码的长度。不能大于 51。

pucSeed

[in] 指向一个包含种子码的缓冲区。种子码的长度不能超过 51 个字节。

pucNewSoPIN

[out] 用来返回新产生的超级用户 PIN 码。长度为 16 个字节。

注解:

根据参数中指定的种子，产生超级密码，用户需要记下新产生的 PIN 码，用户以后的验证。

返回值:

ET_SUCCESS: 表示成功。
ET_HARD_ERROR: 硬件错误
ET_INVALID_PARAMETER: 无效的参数。
ET_ACCESS_DENY: 权限不够。
ET_COMMUNICATIONS_ERROR: 没有打开设备。

ResetPIN

函数 **ResetPIN** 重新设置普通用户密码为 16 个 ‘F’，相当于解锁。

函数原型:

HRESULT

```
ResetPIN(  
    [in] BYTE* pucSoPIN  
);
```

要求:

OpenToken。

参数:

pucSoPIN

[in] 指向一个包含超级用户 PIN 码的缓冲区。16 个字节长。

注解:

重新设置普通用户密码为 16 个 ‘F’，相当于解锁。命令执行成功后，当前安全状态变成超级用户状态。如果验证超级 PIN 码错误，并且错误值在 0xF0 和 ET_PIN_ERR_MAX (0xFF) 之间，则返回值的后四位为剩余重试的次数。如果返回的是 0xFF，则表示验证 PIN 码出错，且 PIN 码的重试次数不受限制。

返回值:

ET_SUCCESS: 表示成功。

ET_HARD_ERROR: 硬件错误

ET_NOT_SET_PID: 没有设置 PID

ET_INVALID_PARAMETER: 无效的参数。

SetKey

函数 **SetKey** 更新参数指定的密钥，此密钥是用于计算 HMAC—MD5 的。

函数原型:

HRESULT

```
SetKey(  
    [in] Long Keyid,  
    [in] BYTE * pucKeyBuf  
);
```

要求:

OpenToken。

参数:

Keyid

[in] 密钥指示。范围是 1—8。

pucKeyBuf

[in] 指向一个包含密钥的缓冲区。长度为固定的 32 个字节。

注解：

更新参数指定的密钥，此密钥是用于计算 HMAC—MD5 的。其中 KEY 的获得，是通过一个纯软件接口 Soft_MD5HMAC，参见相应说明。密钥长度固定为 32 个字节。

返回值：

ET_SUCCESS：表示成功；
ET_NOT_SET_PID：没有设置 PID
ET_HARD_ERROR：硬件错误
ET_INVALID_PARAMETER：无效的参数；
ET_COMMUNICATIONS_ERROR：没有打开设备。

TurnOnLED

函数 **TurnOnLED** 打开 LED 灯，使其变亮。

函数原型：

HRESULT

TurnOnLED();

要求：

OpenToken。

参数：

无

注解：

必须在普通用户状态或超级用户状态下进行。设备加电后，LED灯是常亮的。

返回值：

ET_SUCCESS：表示成功。
ET_ACCESS_DENY：权限不够。
ET_COMMUNICATIONS_ERROR：没有打开设备。

TurnOffLED

函数 **TurnOffLED** 关闭 LED 灯，使其熄灭。

函数原型：

HRESULT

TurnOffLED();

要求：

OpenToken。

参数：

无

注解：

必须在普通用户状态或超级用户状态下进行。设备加电后，LED灯是常亮的。

返回值：

ET_SUCCESS：表示成功。

ET_ACCESS_DENY：权限不够。

ET_COMMUNICATIONS_ERROR：没有打开设备。

GetSN

函数 GetSN 获得硬件序列号。

函数原型：

HRESULT

GetSN(

[out] BYTE * pucSN

);

要求：

OpenToken。

参数：

pucSN

[out] 用来返回硬件设备的序列号的缓冲区，长度为固定 8 个字节的长度。

注解：

可以在匿名状态下进行。序列号码的长度固定为 8 个字节。

返回值：

ET_SUCCESS：表示成功；

ET_INVALID_PARAMETER：无效的参数；

ET_COMMUNICATIONS_ERROR：没有打开设备。

GenRandom

函数 GenRandom 获得 16 个字节的随机数。

函数原型：

HRESULT

```
GetSN(  
    [out] BYTE* pucRandBuf  
);
```

要求：
OpenToken。

参数：
pucRandBuf
[out] 用来返回 16 个字节的随机数的缓冲区指针。

注解：
用来产生 16 个字节的随机数，如果用户需要更长的随机数，则需要多次调用。

返回值：
ET_SUCCESS：表示成功。
ET_NOT_SET_PID：没有设置 PID
ET_INVALID_PARAMETER：无效的参数。
ET_ACCESS_DENY：权限不够。
ET_COMMUNICATIONS_ERROR：没有打开设备。

GenPid

函数 **GenPid** 根据参数中指定的种子，产生产品标识。

函数原型：

```
HRESULT  
GenPid(  
    [in] Long   SeedLen,  
    [in] BYTE * pucSeed,  
    [out] BYTE * pid  
);
```

要求：
OpenToken。

参数：
SeedLen
[in] 种子码的长度。不能大于 51。
pucSeed
[in] 指向一个包含种子码的缓冲区。不能超过 51 个字节。
pid
[out] 用来返回产生的产品标识。8 个字节长。

注解:

出厂设置的产品标识是 32 位的 ‘1’，用户指定种子，产生新的产品标识。

返回值:

ET_SUCCESS: 表示成功;

ET_HARD_ERROR: 硬件错误

ET_INVALID_PARAMETER: 无效的参数;

ET_ACCESS_DENY: 权限不够，需要先验证 SOPIN。

ET_COMMUNICATIONS_ERROR: 没有打开设备。

SetupToken

函数 **SetupToken** 对硬件进行配置。

函数原型:

HRESULT

SetupToken(

[in] BYTE bSoPINRetries,

[in] BYTE bUserPINRetries,

[in] BYTE bUserReadOnly,

[in] BYTE bReserved

);

要求:

OpenToken。

参数:

bSoPINRetries

[in] 超级用户 PIN 码的重试次数。

bUserPINRetries

[in] 普通用户 PIN 码的重试次数。

bUserReadOnly

[in] 用户对数据区的读写标志。

ET_USER_WRITE_READ 可读写

ET_USER_READ_ONLY 只允许读

bReserved

[in] 保留值，必须为 0。

注解:

必须在超级用户状态下进行。

返回值:

ET_SUCCESS: 表示成功。

ET_HARD_ERROR: 硬件错误
ET_INVALID_PARAMETER: 无效的参数。
ET_ACCESS_DENY: 权限不够。
ET_COMMUNICATIONS_ERROR: 没有打开设备。

Read

函数 **Read** 从指定的位置从数据区中读取指定的数据长度。

函数原型:

HRESULT

```
Read(  
    [in] Long    offset,  
    [in] Long    IBytesToRead,  
    [out] BYTE * pucReadBuf  
);
```

要求:

OpenToken。

参数:

lOffset

[in] 相对于起始位置的偏移量。

IBytesToRead

[in] 需要读取的数据长度。不能超过 60 个字节，如果超过，则需要多次调用。

pucReadBuf

[out,] 一个用来返回读取数据的缓冲区指针。长度不能小于 IBytesToRead

注解:

此函数调用需要有普通用户权限。

返回值:

ET_SUCCESS: 表示成功。
ET_INVALID_PARAMETER: 无效的参数。
ET_NOT_SET_PID: 没有设置 PID。
ET_ACCESS_DENY: 权限不够。
ET_COMMUNICATIONS_ERROR: 没有打开设备。

Write

函数 **Write** 将指定的数据长度写到数据区的指定位置。

函数原型:

HRESULT

```
Write(  
    [in] Long    offset,  
    [in] Long    lBytesToRead,  
    [in] BYTE * pucWriteBuf  
);
```

要求:

OpenToken。

参数:

offset

[in] 相对于起始位置的偏移量。

lBytesToRead

[in] 写入数据长度。不能超过 60 个字节，如果超过，则需要多次调用。

pucWriteBuf

[in] 指向一个包含写入数据的缓冲区。

注解:

此函数有存取权限控制。

返回值:

ET_SUCCESS: 表示成功。

ET_HARD_ERROR: 硬件错误

ET_INVALID_PARAMETER: 无效的参数。

ET_NOT_SET_PID: 没有设置 PID。

ET_ACCESS_DENY: 权限不够。

ET_COMMUNICATIONS_ERROR: 没有打开设备。

Soft_MD5HMAC

函数 **Soft_MD5HMAC** 是标准 HMAC_MD5 的软件实现。

函数原型:

HRESULT

```
Soft_MD5HMAC(  
    [in] BYTE * pucText,  
    [in] Long  ulText_Len,  
    [in] BYTE * pucKey,  
    [in] Long  ulKey_Len,  
    [out] BYTE * pucToenKey,  
    [out] BYTE * pucDigest  
);
```

参数:

pucText

[in] 等处理的数据缓存区指针，为空或者范围：大于等于1小于等于51个字节。

ulText_Len

[in] 当pucText不为空时，大于等于1小于等于51。

pucKey

[in] 密钥，按标准RFC2104，长度可以任意

ulKey_Len

[in] 密钥长度

pucToenKey

[out] 硬件计算需要的KEY，固定32字节。

PucDigest

[out] 散列计算的结果，固定 16 字节。

返回值:

ET_SUCCESS: 表示成功。

ET_INVALID_PARAMETER: 无效的参数。

MD5HMAC

函数 **MD5HMAC** 是利用硬件计算 HMAC-MD5。

函数原型:

HRESULT

MD5HMAC(

[in] Long keyID,

[in] Long textLen,

[in] BYTE * pucText,

[out] BYTE *digest

);

要求:

OpenToken。

参数:

lKeyID

[in] 密钥指示。范围是 1—8。

textLen

[in] 待计算的数据，不能大于 51 和小于 1。

pucText

[in] 待计算的数据，长度不能超过51个字节和小于1个字节。

digest

[out] 散列计算的结果，固定 16 字节。

注解：

权限等同于 **KEY** 的读权限。

返回值：

ET_SUCCESS：表示成功；

ET_NOT_SET_PID：没有设置 PID；

ET_INVALID_PARAMETER：无效的参数；

ET_COMMUNICATIONS_ERROR：没有打开设备。

常量定义

常量	值
ET_VERIFY_USERPIN	0x00
ET_VERIFY_SOPIN	0x01

常量	值
ET_USER_WRITE_READ	0x00
ET_USER_READ_ONLY	0x01

常量	描述	值
ET_PIN_ERR_MASK	验证 PIN 码掩码	0x0F
ET_PIN_ERR_MAX	验证 PIN 码最大剩余次数	0xFF

错误代码值

返回状态	描述	值
ET_SUCCESS	函数执行成功	0x00
ET_ACCESS_DENY	访问被拒绝，权限不够	0x01
ET_COMMUNICATIONS_ERROR	通讯错误，没有打开设备	0x02
ET_INVALID_PARAMETER	无效的参数，参数出错	0x03
ET_NOT_SET_PID	没有设置 PID	0x04
ET_UNIT_NOT_FOUND	打开指定的设备失败	0x05
ET_HARD_ERROR	硬件错误	0x06
ET_UNKNOWN_ERROR	未知错误	0x07