

# Re: CryptAPI

---

*Source:* <http://www.tech-archive.net/Archive/VB/microsoft.public.vb.winapi/2005-10/msg00222.html>

---

- *From:* "Tony Proctor" <[tony\\_proctor@xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx](mailto:tony_proctor@xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)>
  - *Date:* Tue, 25 Oct 2005 09:47:57 +0100
- 

You might find this previous thread of interest Michel:

<http://groups.google.ie/group/microsoft.public.vb.general.discussion/msg/548aac4f1ecdc04c?hl=en>

I spent some time here debugging CryptAPI stuff in a multinational environment. This previous thread summarises my findings and the approach I decided to take.

Tony Proctor

"m.poseth" <[michelp@xxxxxxxxxxxxxxxxxxxx](mailto:michelp@xxxxxxxxxxxxxxxxxxxx)> wrote in message

[news:ubxvEIM2FHA.620@xxxxxxxxxxxxxxxxxxxxxxxxxxxx](mailto:news:ubxvEIM2FHA.620@xxxxxxxxxxxxxxxxxxxxxxxxxxxx)

> From the KLPD API Guide

>

> 'Paste this code in a Class Module, named clsCryptoFilterBox

>

> Option Explicit

>

> Private Declare Function CryptAcquireContext Lib "advapi32.dll" Alias

> "CryptAcquireContextA" (phProv As Long, pszContainer As String,

pszProvider

> As String, ByVal dwProvType As Long, ByVal dwFlags As Long) As Long

>

> Private Declare Function CryptCreateHash Lib "advapi32.dll" (ByVal hProv

As

> Long, ByVal Algid As Long, ByVal hKey As Long, ByVal dwFlags As Long,

phHash

> As Long) As Long

>

> Private Declare Function CryptDeriveKey Lib "advapi32.dll" (ByVal hProv As

> Long, ByVal Algid As Long, ByVal hBaseData As Long, ByVal dwFlags As Long,

> phKey As Long) As Long

>

> Private Declare Function CryptDestroyHash Lib "advapi32.dll" (ByVal hHash

As

> Long) As Long

>

> Private Declare Function CryptDestroyKey Lib "advapi32.dll" (ByVal hKey As

> Long) As Long

>

## Re: CryptAPI

```
> Private Declare Function CryptEncrypt Lib "advapi32.dll" (ByVal hKey As
> Long, ByVal hHash As Long, ByVal Final As Long, ByVal dwFlags As Long,
ByVal
> pbData As String, pdwDataLen As Long, ByVal dwBufLen As Long) As Long
>
> Private Declare Function CryptDecrypt Lib "advapi32.dll" (ByVal hKey As
> Long, ByVal hHash As Long, ByVal Final As Long, ByVal dwFlags As Long,
ByVal
> pbData As String, pdwDataLen As Long) As Long
>
> Private Declare Function CryptExportKey Lib "advapi32.dll" (ByVal hKey As
> Long, ByVal hExpKey As Long, ByVal dwBlobType As Long, ByVal dwFlags As
> Long, ByVal pbData As String, pdwDataLen As Long) As Long
>
> Private Declare Function CryptGenKey Lib "advapi32.dll" (ByVal hProv As
> Long, ByVal AlgId As Long, ByVal dwFlags As Long, phKey As Long) As Long
>
> Private Declare Function CryptGetProvParam Lib "advapi32.dll" (ByVal hProv
> As Long, ByVal dwParam As Long, pbData As Any, pdwDataLen As Long, ByVal
> dwFlags As Long) As Long
>
> Private Declare Function CryptGetUserKey Lib "advapi32.dll" (ByVal hProv
As
> Long, ByVal dwKeySpec As Long, phUserKey As Long) As Long
>
> Private Declare Function CryptHashData Lib "advapi32.dll" (ByVal hHash As
> Long, ByVal pbData As String, ByVal dwDataLen As Long, ByVal dwFlags As
> Long) As Long
>
> Private Declare Function CryptReleaseContext Lib "advapi32.dll" (ByVal
hProv
> As Long, ByVal dwFlags As Long) As Long
>
> Private Declare Function CryptSignHash Lib "advapi32.dll" Alias
> "CryptSignHashA" (ByVal hHash As Long, ByVal dwKeySpec As Long, ByVal
> sDescription As String, ByVal dwFlags As Long, ByVal pbSignature As
String,
> pdwSigLen As Long) As Long
>
> Private Declare Function CryptVerifySignature Lib "advapi32.dll" Alias
> "CryptVerifySignatureA" (ByVal hHash As Long, ByVal pbSignature As String,
> ByVal dwSigLen As Long, ByVal hPubKey As Long, ByVal sDescription As
String,
> ByVal dwFlags As Long) As Long
>
> 'API error function
>
> Private Declare Function GetLastError Lib "kernel32" () As Long
>
> 'API memory functions
>
```

## Re: CryptAPI

```
> Private Declare Function GlobalAlloc Lib "kernel32" (ByVal wFlags As Long,
> ByVal dwBytes As Long) As Long
>
> Private Declare Function GlobalFree Lib "kernel32" (ByVal hMem As Long) As
> Long
>
> Private Declare Function GlobalLock Lib "kernel32" (ByVal hMem As Long) As
> Long
>
> Private Declare Function GlobalUnlock Lib "kernel32" (ByVal hMem As Long)
As
> Long
>
> Private Declare Sub CpyMemValAdrFromRefAdr Lib "kernel32" Alias
> "RtlMoveMemory" (ByVal hpvDest As Any, hpvSource As Any, ByVal cbCopy As
> Long)
>
> Private Declare Sub CpyMemRefAdrFromValAdr Lib "kernel32" Alias
> "RtlMoveMemory" (hpvDest As Any, ByVal hpvSource As Any, ByVal cbCopy As
> Long)
>
> 'constants for API memory functions
>
> Private Const GMEM_MOVEABLE = &H2
>
> Private Const GMEM_ZEROINIT = &H40
>
> Private Const GHND = (GMEM_MOVEABLE Or GMEM_ZEROINIT)
>
> 'constants for Cryptography API functions
>
> Private Const MS_DEF_PROV = "Microsoft Base Cryptographic Provider v1.0"
>
> Private Const PROV_RSA_FULL = 1
>
> Private Const CRYPT_NEWKEYSET = &H8
>
> Private Const PP_CONTAINER = 6
>
> Private Const AT_KEYEXCHANGE = 1
>
> Private Const AT_SIGNATURE = 2
>
> Private Const SIMPLEBLOB = 1
>
> Private Const ALG_CLASS_DATA_ENCRYPT = 24576
>
> Private Const ALG_CLASS_HASH = 32768
>
> Private Const ALG_TYPE_ANY = 0
>
```

Re: CryptAPI

Re: CryptAPI

```
> Private Const ALG_TYPE_BLOCK = 1536
>
> Private Const ALG_TYPE_STREAM = 2048
>
> Private Const ALG_SID_RC2 = 2
>
> Private Const ALG_SID_RC4 = 1
>
> Private Const ALG_SID_MD5 = 3
>
> Private Const CALG_MD5 = ((ALG_CLASS_HASH Or ALG_TYPE_ANY) Or ALG_SID_MD5)
>
> Private Const CALG_RC2 = ((ALG_CLASS_DATA_ENCRYPT Or ALG_TYPE_BLOCK) Or
> ALG_SID_RC2)
>
> Private Const CALG_RC4 = ((ALG_CLASS_DATA_ENCRYPT Or ALG_TYPE_STREAM) Or
> ALG_SID_RC4)
>
> 'constants from WinErr.h
>
> Private Const NTE_NO_KEY As Long = -2146893811 '0x8009000DL
>
> Private Const NTE_BAD_SIGNATURE As Long = -2146893818
>
> 'clsCryptoFilterBox constants
>
> Private Const CFB_BUSY = 0
>
> Private Const CFB_READY = 1
>
> Private Const CFB_VALID = 2
>
> Private Const ENCRYPT_ALGORITHM = CALG_RC4
>
> Private Const ENCRYPT_BLOCK_SIZE = 1
>
> Private Const CRYPT_EXPORTABLE = 1
>
> 'private property buffers
>
> Private sInBuffer As String
>
> Private sOutBuffer As String
>
> Private sPassword As String
>
> Private sSignature As String
>
> Private lStatus As Long
>
> Public Property Get InBuffer() As String
```

Re: CryptAPI

```
>
> InBuffer = sInBuffer
>
> End Property
>
> Public Property Let InBuffer(vNewValue As String)
>
> sInBuffer = vNewValue
>
> End Property
>
> Public Property Get OutBuffer() As String
>
> OutBuffer = sOutBuffer
>
> End Property
>
> Public Property Get Signature() As String
>
> Signature = sSignature
>
> End Property
>
> Public Property Let Signature(vNewValue As String)
>
> sSignature = vNewValue
>
> End Property
>
> Public Sub Sign()
>
> 'Create a signature for Inbuffer and place in Signature
>
> Dim sContainer As String, sDescription As String, sProvider As String,
> IHCryptprov As Long
>
> Dim lHHash As Long, lResult As Long, lSignatureLen As Long
>
> On Error GoTo ErrSign
>
> 'switch Status property
>
> lStatus = CFB_BUSY
>
> 'init Signature property
>
> sSignature = ""
>
> 'Get handle to the default provider.
>
> sContainer = vbNullChar
```

## Re: CryptAPI

```
>  
> sProvider = MS_DEF_PROV & vbNullChar  
>  
> If Not CBool(CryptAcquireContext(IHCryptprov, ByVal sContainer, ByVal  
> sProvider, PROV_RSA_FULL, 0)) Then  
>  
> MsgBox ("Error " & CStr(GetLastError) & " during CryptAcquireContext!")  
>  
> GoTo ReleaseHandles:  
>  
> End If  
>  
> 'Create a hash object.  
>  
> If Not CBool(CryptCreateHash(IHCryptprov, CALG_MD5, 0, 0, IHHash)) Then  
>  
> MsgBox ("Error " & CStr(GetLastError) & " during CryptCreateHash!")  
>  
> GoTo ReleaseHandles:  
>  
> End If  
>  
> If Not CBool(CryptHashData(IHHash, sInBuffer, Len(sInBuffer), 0)) Then  
>  
> MsgBox ("Error " & CStr(GetLastError) & " during CryptHashData!")  
>  
> GoTo ReleaseHandles:  
>  
> End If  
>  
> 'Sign hash object.  
>  
> 'Determine size of signature.  
>  
> sDescription = vbNullChar  
>  
> lResult = CryptSignHash(IHHash, AT_SIGNATURE, sDescription, 0, sSignature,  
> lSignatureLen)  
>  
> sSignature = String(lSignatureLen, vbNullChar)  
>  
> 'Sign hash object (with signature key).  
>  
> If Not CBool(CryptSignHash(IHHash, AT_SIGNATURE, sDescription, 0,  
> sSignature, lSignatureLen)) Then  
>  
> MsgBox ("Error " & CStr(GetLastError()) & " during CryptSignHash")  
>  
> GoTo ReleaseHandles:  
>  
> End If
```

Re: CryptAPI

Re: CryptAPI

```
>
> ReleaseHandles:
>
> 'Destroy hash object.
>
> If IHHash Then IResult = CryptDestroyHash(IHHash)
>
> 'Release provider handle.
>
> If IHCryptprov Then IResult = CryptReleaseContext(IHCryptprov, 0)
>
> 'switch Status property
>
> IStatus = CFB_READY
>
> Exit Sub
>
> ErrSign:
>
> MsgBox ("ErrSign " & Error$)
>
> GoTo ReleaseHandles
>
> End Sub
>
> Public Sub Validate()
>
> 'Validate InBuffer with Signature and assign Status with result
>
> Dim bValid As Boolean, sContainer As String, sDescription As String,
> sProvider As String
>
> Dim IDataLen As Long, IDataPoint As Long, IHCryptprov As Long, IHHash As
> Long
>
> Dim IResult As Long, ISignatureLen As Long, IHCryptKey As Long
>
> ReDim aByteData(0) As Byte
>
> On Error GoTo ErrValidate
>
> 'switch Status property
>
> IStatus = CFB_BUSY
>
> 'init internal valid flag
>
> bValid = True
>
> 'Get handle to the default provider.
>
```

## Re: CryptAPI

```
> sContainer = vbNullChar
>
> sProvider = MS_DEF_PROV & vbNullChar
>
> If Not CBool(CryptAcquireContext(IHCryptprov, ByVal sContainer, ByVal
> sProvider, PROV_RSA_FULL, 0)) Then
>
> bValid = False
>
> MsgBox ("Error " & CStr(GetLastError) & " during CryptAcquireContext!")
>
> GoTo ReleaseHandles:
>
> End If
>
> 'Create a hash object.
>
> If Not CBool(CryptCreateHash(IHCryptprov, CALG_MD5, 0, 0, IHHash)) Then
>
> bValid = False
>
> MsgBox ("Error " & CStr(GetLastError) & " during CryptCreateHash!")
>
> GoTo ReleaseHandles:
>
> End If
>
> 'Add data to hash object.
>
> If Not CBool(CryptHashData(IHHash, sInBuffer, Len(sInBuffer), 0)) Then
>
> bValid = False
>
> MsgBox ("Error " & CStr(GetLastError) & " during CryptHashData!")
>
> GoTo ReleaseHandles:
>
> End If
>
> 'Determine size of signature.
>
> 'sDescription = vbNullChar
>
> 'lResult = CryptSignHash(IHHash, AT_SIGNATURE, sDescription, 0, 0,
> lSignatureLen)
>
> 'Get handle to signature key.
>
> If Not CBool(CryptGetUserKey(IHCryptprov, AT_SIGNATURE, IHCryptKey)) Then
>
> bValid = False
```

Re: CryptAPI

## Re: CryptAPI

```
>
> MsgBox ("Error " & CStr(GetLastError) & " during CryptGetUserKey!")
>
> GoTo ReleaseHandles:
>
> End If
>
> lSignatureLen = Len(sSignature)
>
> 'Verify signature.
>
> If Not CBool(CryptVerifySignature(lHHash, sSignature, lSignatureLen,
> lHCryptKey, sDescription, 0)) Then
>
> If GetLastError = NTE_BAD_SIGNATURE Then
>
> bValid = False
>
> GoTo ReleaseHandles:
>
> Else
>
> bValid = False
>
> MsgBox ("Error " & CStr(GetLastError) & " during CryptVerifySignature!")
>
> GoTo ReleaseHandles:
>
> End If
>
> End If
>
> ReleaseHandles:
>
> 'Release signature key.
>
> If lHCryptKey Then lResult = CryptDestroyKey(lHCryptKey)
>
> 'Destroy hash object.
>
> If lHHash Then lResult = CryptDestroyHash(lHHash)
>
> 'Release provider handle.
>
> If lHCryptprov Then lResult = CryptReleaseContext(lHCryptprov, 0)
>
> Select Case bValid
>
> Case True
>
> lStatus = CFB_VALID
```

Re: CryptAPI

Re: CryptAPI

```
>
> Case Else
>
> IStatus = CFB_READY
>
> End Select
>
> Exit Sub
>
> ErrValidate:
>
> MsgBox ("ErrValidate " & Error$)
>
> Resume
>
> End Sub
>
> Public Sub Encrypt()
>
> 'Encrypt InBuffer into OutBuffer
>
> Dim IHExchgKey As Long, IHCryptprov As Long, IHHash As Long, IHkey As Long
>
> Dim IResult As Long, sContainer As String, sProvider As String,
sCryptBuffer
> As String
>
> Dim ICryptLength As Long, ICryptBufLen As Long
>
> On Error GoTo ErrEncrypt
>
> 'switch Status property
>
> IStatus = CFB_BUSY
>
> 'Get handle to the default provider
>
> sContainer = vbNullChar
>
> sProvider = vbNullChar
>
> sProvider = MS_DEF_PROV & vbNullChar
>
> If Not CBool(CryptAcquireContext(IHCryptprov, ByVal sContainer, ByVal
> sProvider, PROV_RSA_FULL, 0)) Then
>
> MsgBox ("Error " & CStr(GetLastError) & " during CryptAcquireContext!")
>
> GoTo Done
>
> End If
```

## Re: CryptAPI

```
>  
> 'Create a hash object.  
>  
> If Not CBool(CryptCreateHash(IHCryptprov, CALG_MD5, 0, 0, IHHash)) Then  
>  
> MsgBox ("Error " & CStr(GetLastError) & " during CryptCreateHash!")  
>  
> GoTo Done  
>  
> End If  
>  
> 'Hash in the password data.  
>  
> If Not CBool(CryptHashData(IHHash, sPassword, Len(sPassword), 0)) Then  
>  
> MsgBox ("Error " & CStr(GetLastError) & " during CryptHashData!")  
>  
> GoTo Done  
>  
> End If  
>  
> 'Derive a session key from the hash object.  
>  
> If Not CBool(CryptDeriveKey(IHCryptprov, ENCRYPT_ALGORITHM, IHHash, 0,  
> IHkey)) Then  
>  
> MsgBox ("Error " & CStr(GetLastError) & " during CryptDeriveKey!")  
>  
> GoTo Done  
>  
> End If  
>  
> 'Destroy the hash object.  
>  
> CryptDestroyHash (IHHash)  
>  
> IHHash = 0  
>  
> 'Prepare a string buffer for the CryptEncrypt function  
>  
> lCryptLength = Len(sInBuffer)  
>  
> lCryptBufLen = lCryptLength * 2  
>  
> sCryptBuffer = String(lCryptBufLen, vbNullChar)  
>  
> LSet sCryptBuffer = sInBuffer  
>  
> 'Encrypt data  
>  
> If Not CBool(CryptEncrypt(IHkey, 0, 1, 0, sCryptBuffer, lCryptLength,
```

## Re: CryptAPI

```
> lCryptBufLen)) Then
>
> MsgBox ("bytes required:" & CStr(lCryptLength))
>
> MsgBox ("Error " & CStr(GetLastError) & " during CryptEncrypt!")
>
> 'GoTo Done
>
> End If
>
> sOutBuffer = Mid$(sCryptBuffer, 1, lCryptLength)
>
> Done:
>
> 'Destroy session key.
>
> If (IHkey) Then lResult = CryptDestroyKey(IHkey)
>
> 'Release key exchange key handle.
>
> If IHExchgKey Then CryptDestroyKey (IHExchgKey)
>
> 'Destroy hash object.
>
> If IHHash Then CryptDestroyHash (IHHash)
>
> 'Release provider handle.
>
> If IHCryptprov Then lResult = CryptReleaseContext(IHCryptprov, 0)
>
> 'switch Status property
>
> lStatus = CFB_READY
>
> Exit Sub
>
> ErrEncrypt:
>
> MsgBox ("ErrEncrypt " & Error$)
>
> Resume
>
> End Sub
>
> Public Sub Decrypt()
>
> 'Decrypt InBuffer into OutBuffer
>
> Dim IHExchgKey As Long, IHCryptprov As Long, IHHash As Long, IHkey As Long
>
> Dim lResult As Long, sContainer As String, sProvider As String
```

Re: CryptAPI

## Re: CryptAPI

```
>
> Dim sCryptBuffer As String, lCryptBufLen As Long, lCryptPoint As Long
>
> Dim lPasswordPoint As Long, lPasswordCount As Long
>
> On Error GoTo ErrDecrypt
>
> 'switch Status property
>
> lStatus = CFB_BUSY
>
> 'Init sOutBuffer
>
> sOutBuffer = ""
>
> 'Get handle to the default provider.
>
> sContainer = vbNullChar
>
> sProvider = vbNullChar
>
> sProvider = MS_DEF_PROV & vbNullChar
>
> If Not CBool(CryptAcquireContext(lHCryptprov, ByVal sContainer, ByVal
> sProvider, PROV_RSA_FULL, 0)) Then
>
> MsgBox ("Error " & CStr(GetLastError) & " during CryptAcquireContext!")
>
> GoTo Done
>
> End If
>
> 'Create a hash object.
>
> If Not CBool(CryptCreateHash(lHCryptprov, CALG_MD5, 0, 0, lHHash)) Then
>
> MsgBox ("Error " & CStr(GetLastError) & " during CryptCreateHash!")
>
> GoTo Done
>
> End If
>
> 'Hash in the password data.
>
> If Not CBool(CryptHashData(lHHash, sPassword, Len(sPassword), 0)) Then
>
> MsgBox ("Error " & CStr(GetLastError) & " during CryptHashData!")
>
> GoTo Done
>
> End If
```

Re: CryptAPI

## Re: CryptAPI

```
>
> 'Derive a session key from the hash object.
>
> If Not CBool(CryptDeriveKey(IHCryptprov, ENCRYPT_ALGORITHM, IHHash, 0,
> IHkey)) Then
>
> MsgBox ("Error " & CStr(GetLastError) & " during CryptDeriveKey!")
>
> GoTo Done
>
> End If
>
> 'Destroy the hash object.
>
> CryptDestroyHash (IHHash)
>
> IHHash = 0
>
> 'Prepare sCryptBuffer for CryptDecrypt
>
> lCryptBufLen = Len(sInBuffer) * 2
>
> sCryptBuffer = String(lCryptBufLen, vbNullChar)
>
> LSet sCryptBuffer = sInBuffer
>
> 'Decrypt data
>
> If Not CBool(CryptDecrypt(IHkey, 0, 1, 0, sCryptBuffer, lCryptBufLen))
Then
>
> MsgBox ("bytes required:" & CStr(lCryptBufLen))
>
> MsgBox ("Error " & CStr(GetLastError) & " during CryptDecrypt!")
>
> GoTo Done
>
> End If
>
> 'Apply decrypted string from sCryptBuffer to private buffer for OutBuffer
> property
>
> sOutBuffer = Mid$(sCryptBuffer, 1, Len(sInBuffer))
>
> Done:
>
> 'Destroy session key.
>
> If (IHkey) Then lResult = CryptDestroyKey(IHkey)
>
> 'Release key exchange key handle.
```

Re: CryptAPI

Re: CryptAPI

```
>
> If IHExchgKey Then CryptDestroyKey (IHExchgKey)
>
> 'Destroy hash object.
>
> If IHHash Then CryptDestroyHash (IHHash)
>
> 'Release provider handle.
>
> If IHCryptprov Then lResult = CryptReleaseContext(IHCryptprov, 0)
>
> 'switch Status property
>
> lStatus = CFB_READY
>
> Exit Sub
>
> ErrDecrypt:
>
> MsgBox ("ErrDecrypt " & Error$)
>
> GoTo Done
>
> End Sub
>
> Public Property Get Status() As Long
>
> Status = lStatus
>
> End Property
>
> Private Function InitUser() As Long
>
> Dim IHCryptprov As Long, IHCryptKey As Long, avProviderData(1000) As Byte
>
> Dim lProviderDataAddress As Long, lProviderDataLen As Long, lDataSize As
> Long
>
> Dim lResult As Long, sContainer As String, sProvider As String
>
> Dim sUserName As String, lPoint As Long, lMemHandle As Long
>
> Dim lReturn As Long, sBuffer As String
>
> On Error GoTo ErrInitUser
>
> 'prepare string buffers
>
> sContainer = vbNullChar
>
> sProvider = MS_DEF_PROV & vbNullChar
```

## Re: CryptAPI

```
>
> 'Attempt to acquire a handle to the default key container.
>
> If Not CBool(CryptAcquireContext(IHCryptprov, ByVal sContainer, ByVal
> sProvider, PROV_RSA_FULL, 0)) Then
>
> 'Create default key container.
>
> If Not CBool(CryptAcquireContext(IHCryptprov, ByVal sContainer, ByVal
> sProvider, PROV_RSA_FULL, CRYPT_NEWKEYSET)) Then
>
> MsgBox ("Error creating key container! " & CStr(GetLastError))
>
> Exit Function
>
> End If
>
> 'Get name of default key container.
>
> lProviderDataLen = Len(avProviderData(0)) * (UBound(avProviderData) + 1)
>
> If Not CBool(CryptGetProvParam(IHCryptprov, PP_CONTAINER,
avProviderData(0),
> lProviderDataLen, 0)) Then
>
> MsgBox ("Error getting user name! " & CStr(GetLastError))
>
> avProviderData(0) = 0
>
> End If
>
> 'Get sUserName from avProviderData()
>
> lPoint = LBound(avProviderData)
>
> While lPoint <= UBound(avProviderData)
>
> If avProviderData(lPoint) <> 0 Then
>
> sUserName = sUserName & Chr$(avProviderData(lPoint))
>
> Else
>
> lPoint = UBound(avProviderData)
>
> End If
>
> lPoint = lPoint + 1
>
> Wend
>
```

## Re: CryptAPI

```
> MsgBox ("Create key container " & sUserName)
>
> End If
>
> 'Attempt to get handle to signature key
>
> If Not CBool(CryptGetUserKey(IHCryptprov, AT_SIGNATURE, IHCryptKey)) Then
>
> If GetLastError = NTE_NO_KEY Then
>
> MsgBox ("Create key exchange key pair")
>
> If Not CBool(CryptGenKey(IHCryptprov, AT_SIGNATURE, 0, IHCryptKey)) Then
>
> MsgBox ("Error during CryptGenKey! " & CStr(GetLastError))
>
> Exit Function
>
> Else
>
> IResult = CryptDestroyKey(IHCryptprov)
>
> End If
>
> Else
>
> MsgBox ("Error during CryptGetUserKey! " & CStr(GetLastError))
>
> Exit Function
>
> End If
>
> End If
>
> 'Attempt to get handle to exchange key
>
> If Not CBool(CryptGetUserKey(IHCryptprov, AT_KEYEXCHANGE, IHCryptKey))
Then
>
> If GetLastError = NTE_NO_KEY Then
>
> MsgBox ("Create key exchange key pair")
>
> If Not CBool(CryptGenKey(IHCryptprov, AT_KEYEXCHANGE, 0, IHCryptKey)) Then
>
> MsgBox ("Error during CryptGenKey! " & CStr(GetLastError))
>
> Exit Function
>
> Else
>
>
```

Re: CryptAPI

Re: CryptAPI

```
> IResult = CryptDestroyKey(IHCryptprov)
>
> End If
>
> Else
>
> MsgBox ("Error during CryptGetUserKey! " & CStr(GetLastError))
>
> Exit Function
>
> End If
>
> End If
>
> 'release handle to provider
>
> IResult = CryptReleaseContext(IHCryptprov, 0)
>
> InitUser = True
>
> Exit Function
>
> ErrInitUser:
>
> MsgBox ("ErrInitUser " & Error$)
>
> Resume
>
> End Function
>
> Private Sub Class_Initialize()
>
> If InitUser = True Then
>
> MsgBox ("InitUser OK")
>
> Else
>
> MsgBox ("InitUser failed")
>
> End If
>
> End Sub
>
> Public Property Get Password() As String
>
> Password = sPassword
>
> End Property
>
> Public Property Let Password(vNewValue As String)
```

Re: CryptAPI

>  
> sPassword = vNewValue  
>  
> End Property  
>  
>  
> regards  
>  
> Michel Posseth [MCP]  
>  
>  
>  
>  
>  
>  
>  
>  
> "Robertico" <Robertico@xxxxxxxxxxxxxxxx> wrote in message  
> [news:djd731\\$yno\\$1@xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx](mailto:news:djd731$yno$1@xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)  
>> Looking for an example using the CryptAPI.  
>> Want to add encryption to a chat program.  
>>  
>> Thanks in advance,  
>>  
>> Robertico  
>>  
>  
>

---

• *Follow-Ups:*

- ◆ [Re: CryptAPI](#)  
◇ From: Robertico

• *References:*

- ◆ [CryptAPI](#)  
◇ From: Robertico
- ◆ [Re: CryptAPI](#)  
◇ From: m.posseth

- Prev by Date: [Re: FlatSB SetScrollPos](#)
- Next by Date: [Re: using serial port using APIs](#)
- Previous by thread: [Re: CryptAPI](#)
- Next by thread: [Re: CryptAPI](#)
- Index(es):
  - ◆ [Date](#)
  - ◆ [Thread](#)