

# LNCAToolkits 控件(通用版)程序员手册 v2.5.0.1



辽宁 CA 版权所有(2005)



地址：沈阳市和平区和平北大街 28 甲 3（110006）

TEL: 024-23210366

FAX: 024-23871490

[Http://www.lnca.org.cn](http://www.lnca.org.cn)

## 目 录

<b>1</b>	<b>概述 .....</b>	<b>6</b>
<b>2</b>	<b>功能 .....</b>	<b>6</b>
<b>3</b>	<b>注意事项 .....</b>	<b>7</b>
<b>4</b>	<b>算法说明 .....</b>	<b>7</b>
<b>5</b>	<b>使用方法 .....</b>	<b>8</b>
5.1	网页中调用 .....	8
5.2	VC中调用 .....	8
5.3	VB中调用 .....	9
<b>6</b>	<b>属性 .....</b>	<b>9</b>
6.1	属性简要描述 .....	9
6.2	客户端签名证书属性 .....	10
	<i>ClientSignCertSN</i> //吉大兼容 .....	10
	<i>ClientSignCertCN</i> //吉大兼容 .....	10
	<i>ClientSignCertDN</i> .....	10
	<i>ClientSignCertIssuer</i> .....	11
	<i>ClientSignCert</i> //吉大兼容 .....	11
6.3	客户端签名属性 .....	11
	<i>ClientEncCertSN</i> .....	11
	<i>ClientEncCertCN</i> .....	11
	<i>ClientEncCertDN</i> .....	11
	<i>ClientEncCertIssuer</i> .....	11
	<i>ClientEncCert</i> //吉大兼容 .....	11
6.4	其它属性 .....	12
	<i>ErrorCode</i> //吉大兼容 .....	12
	<i>ErrorMessage</i> //吉大兼容 .....	12
<b>7</b>	<b>方法 .....</b>	<b>12</b>
7.1	证书初始化方法 .....	12
	<i>EnumClientCert</i> .....	12
	<i>InitClientCert</i> //吉大兼容 .....	12

<i>InitClientCertByCertSN</i> //吉大兼容.....	13
<i>InitClientCertByCertDN</i> .....	13
<i>InitServerCertFromStore</i> .....	13
<i>InitThirdCertFromStore</i> .....	13
<i>InitServerCert</i> //吉大兼容 .....	13
<i>InitThirdCert</i> .....	14
<i>RemoveCert</i> .....	14
<i>CertInitStatus</i> .....	14
7.2 证书解析方法.....	14
<i>ReadCertSN</i> .....	14
<i>ReadCertCN</i> .....	15
<i>ReadCertDN</i> .....	15
<i>ReadCertIssuer</i> .....	15
<i>ReadCert</i> .....	15
<i>ReadCertBeginValidityTime</i> .....	16
<i>ReadCertEndValidityTime</i> .....	16
<i>ReadCertExtension</i> .....	16
<i>ReadCertExtensionCount</i> .....	17
<i>ReadCertExtensionOID</i> .....	17
<i>ReadCertExtensionValue</i> .....	17
<i>ReadCertProviderName</i> .....	18
<i>CompareCertProviderName</i> .....	18
7.3 证书主题解析方法.....	18
<i>ReadCertDN_OU</i> .....	18
<i>ReadCertDN_OUa</i> .....	18
<i>ReadCertDN_OUi</i> .....	19
<i>ReadCertDN_O</i> .....	19
<i>ReadCertDN_L</i> .....	19
<i>ReadCertDN_S</i> .....	19
<i>ReadCertDN_C</i> .....	20
<i>ReadCertDN_E</i> .....	20
7.4 证书查看.....	20
<i>ViewCertWnd</i> .....	20
7.5 CRL操作.....	20
<i>InitCRL</i> .....	20
<i>VerifyCRL</i> .....	21
<i>CRLVerifyCert</i> .....	21
7.6 哈希操作方法.....	22
<i>HashData</i> .....	22
<i>VerifyHashData</i> .....	22
7.7 签名操作方法.....	22
<i>SignHash</i> .....	22
<i>VerifySignHash</i> .....	23
7.8 PKCS7 签名数据操作方法.....	23

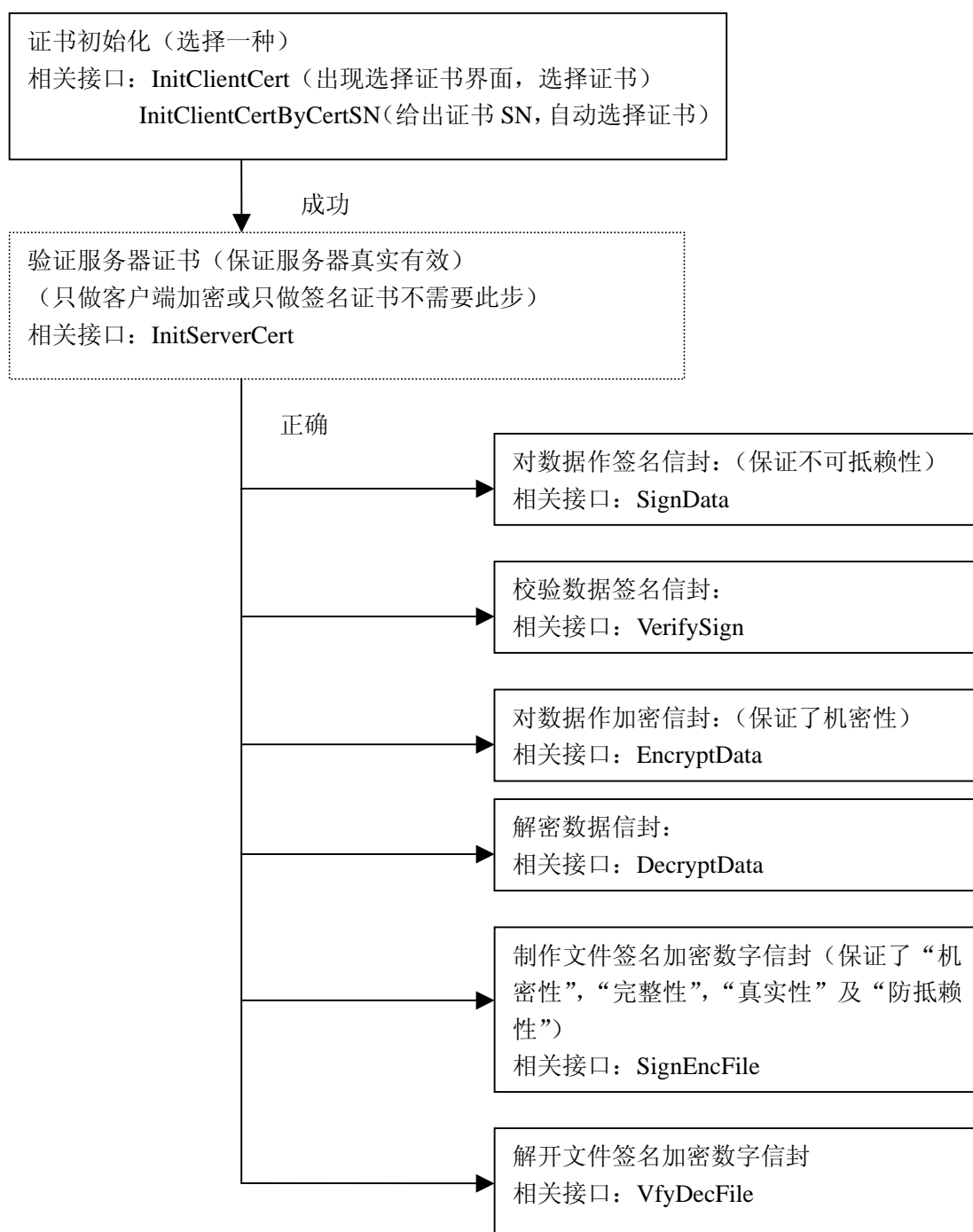
<i>SignDataEx</i> .....	23
<i>SignData</i> // 吉大兼容 .....	24
<i>VerifySignEx</i> .....	24
<i>VerifySign</i> // 吉大兼容 .....	25
<i>SignFileEx</i> .....	25
<i>VerifySignFileEx</i> .....	26
<i>ReturnSignerCertData</i> .....	26
<i>ReturnSignerCertSN</i> .....	26
<i>ReturnSigningTime</i> .....	27
7.9   PKCS7 加密信封操作方法 .....	27
<i>EncryptDataEx</i> .....	27
<i>EncryptData</i> // 吉大兼容 .....	28
<i>DecryptData</i> // 吉大兼容 .....	28
<i>EncryptFileEx</i> .....	28
<i>DecryptFile</i> .....	29
7.10  PKCS7 签名加密信封操作方法 .....	29
<i>SignEnvDataEx</i> .....	29
<i>SignEnvData</i> // 吉大兼容 .....	30
<i>VfyDecDataEx</i> .....	30
<i>VfyDecData</i> // 吉大兼容 .....	31
<i>SignEncFileEx</i> .....	31
<i>SignEncFile</i> // 吉大兼容 .....	32
<i>VfyDecFileEx</i> .....	33
<i>VfyDecFile</i> // 吉大兼容 .....	33
7.11  对称加解密操作方法 .....	33
<i>SymCipherInit</i> .....	33
<i>SymEncryptData</i> .....	34
<i>SymDecryptData</i> .....	34
7.12  LDAP 相关操作方法 .....	34
<i>ReadRootCertFromLdap</i> .....	34
<i>ReadCRLFromLdap</i> .....	35
<i>ReadCertFromLdap</i> .....	35
<i>ReturnCertType</i> .....	35
<i>ReturnCertStatus</i> .....	36
<i>ReturnCertTypeStr</i> .....	36
<i>ReturnCertStatusStr</i> .....	37
7.13  时间戳操作方法 .....	37
<i>SetTimeStampServerUrl</i> .....	37
<i>AppendTimeStampServerUrl</i> .....	38
<i>RequestTimeStamp</i> .....	38
<i>GetTimeStampVersion</i> .....	38
<i>GetTimeStampTimeStr</i> .....	38
<i>GetTimeStampSerialNumber</i> .....	38
<i>GetTimeStampMessageImprint</i> .....	39

<i>GetTimeStampToken</i> .....	39
<i>GetTimeStampSrcHash</i> .....	39
<i>GetTimeStampSrcAlgo</i> .....	39
<i>ViewTimeStampCertWnd</i> .....	40
7.14 其他公共方法.....	40
<i>GenUUID</i> .....	40
<i>CryptGenRandom</i> .....	40
<i>BinToBase64</i> .....	40
<i>Base64ToBin</i> .....	41
<i>ReadDataLength</i> .....	41
<i>StringToByteLength</i> .....	41
<i>StringToUnicodeLength</i> .....	41
<i>AboutBox</i> //吉大兼容.....	42
<i>GetVersion</i> .....	42
<i>SetShowError</i> .....	42
<b>8 返回码与错误信息</b> .....	<b>42</b>

## 1 概述

LNCA Toolkits ActiveX 控件,适用于客户端对证书的各种操作,使用方便,只需要注册一次控件,就可以随时调用和使用。简化了编程操作。

## 2 功能



### 3 注意事项

- 签名操作使用 RSA\_MD5 算法。
- 对称密钥产生使用 3DES 算法。
- 某些函数说明中有“吉大兼容”，即吉大的控件中有相同的功能。
- 本控件内部维护着一个证书数组，初始时数组中的每个元素为空，当成功地初始化证书后其中的数组元素中保存证书对象。其中的编号顺序为：

0 = 客户签名证书  
1 = 客户加密证书  
2 = 服务器签名证书  
3 = 服务器加密证书  
4 = 第三方签名证书  
5 = 第三方加密证书  
6 = 返回签名信封中的签名者证书  
7 = 返回加密信封中的接收人证书  
8 = 根证书  
9 = 时间戳包含的证书

### 4 算法说明

摘要算法：

szOID\_OIWSEC\_sha1 = "1.3.14.3.2.26"  
szOID\_RSA\_MD5 = "1.2.840.113549.2.5"  
szOID\_RSA\_MD4 = "1.2.840.113549.2.4"  
szOID\_RSA\_MD2 = "1.2.840.113549.2.2"

签名算法：

szOID\_OIWSEC\_sha1 = "1.3.14.3.2.26"  
szOID\_RSA\_MD5 = "1.2.840.113549.2.5" (默认算法)  
szOID\_RSA\_MD2 = "1.2.840.113549.2.2"

RSA 加解密算法：

szOID\_RSA\_DES\_EDE3\_CBC = "1.2.840.113549.3.7" (默认算法)  
szOID\_OIWSEC\_desCBC = "1.3.14.3.2.7"  
szOID\_RSA\_RC4 = "1.2.840.113549.3.4"  
szOID\_RSA\_RC2CBC = "1.2.840.113549.3.2"

对称加解密算法：

CALG\_3DES = "1.2.840.113549.3.7" = szOID\_RSA\_DES\_EDE3\_CBC

CALG_DES	= "1.3.14.3.2.7"	= szOID_OIWSEC_desCBC
CALG_RC4	= "1.2.840.113549.3.4"	= szOID_RSA_RC4
CALG_RC2	= "1.2.840.113549.3.2"	= szOID_RSA_RC2CBC

## 5 使用方法

首先必须先注册 LNCAActiveX 控件组件。方法：开始→运行→regsvr32 “控件或组件的绝对路径”。



### 5.1 网页中调用

声明控件：

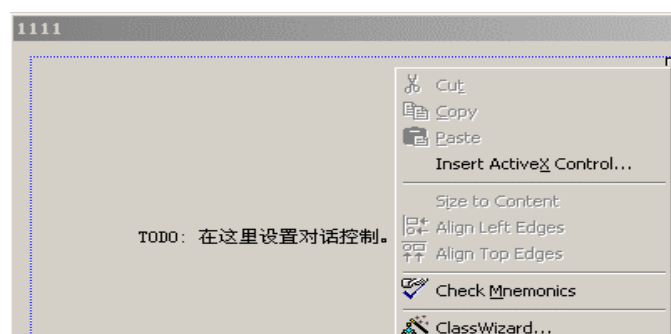
```
<OBJECT id="LNCAToolkits1"
  style="LEFT: 0px; WIDTH: 1px; TOP: 0px; HEIGHT: 1px"
  height="1" width="1"
  classid="clsid: 6FBE853D-0DB0-4C62-B7DC-B49E9D447AF9"
  CODEBASE="http://172.16.1.1/ LNCAToolkits.cab#version=2,5,0,1">
</OBJECT>
```

调用控件接口：

```
parent.TopFrame.document. LNCAToolkits1.InitClientCert
```

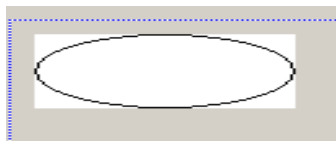
### 5.2 VC 中调用

新建 dialog 工程，再设置对话框控制窗口，单击右键选择 Insert ActiveX Control





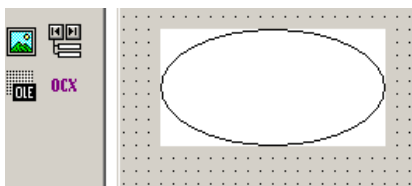
选择相应的控件名称后出现如下控件。设置 ID 后，设置变量值例如 m\_LNCAToolkits。



在程序中利用 m\_LNCAToolkits.InitClientCert();等调用控件接口。

## 5.3 VB 中调用

打开新工程，在工具条上空白处单击右键。选择 Components 再出现的对话框中选择相应控件工具条上出现 ocx 工具，单击此工具在 form 上拖动出现控件。设置控件名称 LNCAToolkits，在程序中就可以利用 LNCAToolkits.InitClientCert();等调用控件接口。



## 6 属性

这个控件提供的属性仅具有只读变量的功能。用户在使用此属性前，必须先初始化客户端证书。它提供以下功能属性。

### 6.1 属性简要描述

- ClientSignCertSN //吉大兼容  
客户端签名证书的序列号。
- ClientSignCertCN //吉大兼容  
客户端签名证书的 CommonName。
- ClientSignCertDN  
客户端签名证书的主题。

- **ClientSignCertIssuer**  
客户端签名证书的颁发者。
- **ClientSignCert** //吉大兼容  
客户端签名证书的 BASE64 编码。
- **ClientEncCertSN**  
客户端加密证书的序列号。
- **ClientEncCertCN**  
客户端加密证书的 CommonName。
- **ClientEncCertDN**  
客户端签名证书的主题。
- **ClientEncCertIssuer**  
客户端加密证书的颁发者。
- **ClientEncCert** //吉大兼容  
客户端加密证书的 BASE64 编码。
- **ErrorCode** //吉大兼容  
错误码。
- **ErrorMessage** //吉大兼容  
错误信息，保留程序运行的最后一个错误信息。

## 6.2 客户端签名证书属性

### **ClientSignCertSN** //吉大兼容

属性声明: BSTR ClientSignCertSN;

说明: 获取客户端签名证书的序列号

### **ClientSignCertCN** //吉大兼容

属性声明: BSTR ClientSignCertCN;

说明: 获取客户端签名证书的公共名称

### **ClientSignCertDN**

属性声明: BSTR ClientSignCertDN;

说明: 获取客户端签名证书的主题

## ClientSignCertIssuer

属性声明: BSTR ClientSignCertIssuer;

说明: 获取客户端签名证书的颁发者主题

## ClientSignCert //吉大兼容

属性声明: BSTR ClientSignCert;

说明: 获取客户端签名证书的内容(BASE64 编码)

## 6.3 客户端签名属性

### ClientEncCertSN

属性声明: BSTR ClientEncCertSN;

说明: 获取客户端加密证书的序列号

### ClientEncCertCN

属性声明: BSTR ClientEncCertCN;

说明: 获取客户端加密证书的公共名称

### ClientEncCertDN

属性声明: BSTR ClientEncCertDN;

说明: 获取客户端加密证书的主题

### ClientEncCertIssuer

属性声明: BSTR ClientEncCertIssuer;

说明: 获取客户端加密证书的颁发者主题

## ClientEncCert //吉大兼容

属性声明: BSTR ClientEncCert;

说明: 获取客户端加密证书的内容(BASE64 编码)

## 6.4 其它属性

### **ErrorCode** //吉大兼容

属性声明: long ErrorCode;

说明: 获取错误号, 在程序出错的时候, 调用本属性, 0 表示成功; 其他值为最后发生的错误的号码

### **ErrorMessage** //吉大兼容

属性声明: BSTR ErrorMessage;

说明: 获取错误消息, 在程序出错的时候, 调用本属性

## 7 方法

### 7.1 证书初始化方法

#### **EnumClientCert**

函数声明: long EnumClientCert();

说明: 从证书库枚举客户端双证书, 如果证书库中有不是一对双证书, 则显示并选择证书列表, 通过人工进行选择来初始化证书, 并且此证书必须包含私钥。

参数:

返回值: 成功时返回 0,

失败时返回<0, 由 ErrorCode 属性中取错误码, 由 ErrorMessage 属性中取错误信息。

#### **InitClientCert** //吉大兼容

函数声明: long InitClientCert();

说明: 初始化客户端证书, 显示并选择证书列表, 并且此证书必须包含私钥。

参数:

返回值: 成功时返回 0,

失败时返回<0, 由 ErrorCode 属性中取错误码, 由 ErrorMessage 属性中取错误信息。

## InitClientCertByCertSN //吉大兼容

函数声明: long InitClientCertByCertSN(BSTR szSN);

说明: 通过证书序列号初始化客户端证书, 并且此证书必须包含私钥。

参数:

- szSN: 客户端的证书序列号

返回值: 成功时返回 0,

失败时返回<0, 由 ErrorCode 属性中取错误码, 由 ErrorMessage 属性中取错误信息。

## InitClientCertByCertDN

函数声明: long InitClientCertByCertDN(BSTR szDN);

说明: 通过证书主题初始化客户端证书, 并且此证书必须包含私钥。

参数:

- szDN: 客户端的主题

返回值: 成功时返回 0,

失败时返回<0, 由 ErrorCode 属性中取错误码, 由 ErrorMessage 属性中取错误信息。

## InitServerCertFromStore

函数声明: long InitServerCertFromStore();

说明: 从系统证书库中选择服务器(对方)证书

参数:

返回值: 成功时返回 0=初始化加密证书,

失败时返回<0, 由 ErrorCode 属性中取错误码, 由 ErrorMessage 属性中取错误信息。

## InitThirdCertFromStore

函数声明: long InitThirdCertFromStore();

说明: 从系统证书库中选择第三方证书

参数:

返回值: 成功时返回 0=初始化加密证书,

失败时返回<0, 由 ErrorCode 属性中取错误码, 由 ErrorMessage 属性中取错误信息。

## InitServerCert //吉大兼容

函数声明: long InitServerCert(BSTR szSignCert, BSTR szEncCert);

说明: 根据公钥证书数据(BASE64 编码)初始化服务器(对方)证书

参数:

- szSignCert: 服务器(对方)签名证书内容(BASE64 编码)
- szEncCert: 服务器(对方)加密证书内容(BASE64 编码)

**返回值：**成功时返回 0=初始化双证书，1=初始化签名证书，2=初始化加密证书  
失败时返回<0，由 ErrorCode 属性中取错误码，由 ErrorMessage 属性中取错误信息。

## InitThirdCert

**函数声明：** long InitThirdCert(BSTR szSignCert, BSTR szEncCert);

**说明：**根据公钥证书数据(BASE64 编码)初始化第三方证书

**参数：**

- szSignCert: 第三方签名公钥证书内容(BASE64 编码)
- szEncCert: 第三方加密公钥证书内容(BASE64 编码)

**返回值：**成功时返回 0=初始化双证书，1=初始化签名证书，2=初始化加密证书  
失败时返回<0，由 ErrorCode 属性中取错误码，由 ErrorMessage 属性中取错误信息。

## RemoveCert

**函数声明：** long RemoveCert (long nCertIndex);

**说明：**清空数组中的指定证书对象，但是不能清空第 8 项为根证书。

**参数：**

- nCertIndex: 输入证书数组序号

**返回值：**成功时返回 0，  
失败时返回<0，由 ErrorCode 属性中取错误码，由 ErrorMessage 属性中取错误信息。

## CertInitStatus

**函数声明：** long CertInitStatus (long nCertIndex);

**说明：**判断数组中的指定证书对象是否已经初始化。

**参数：**

- nCertIndex: 输入证书数组序号

**返回值：**返回 0 = 证书已经初始化，  
返回<0 = 证书未初始化，由 ErrorCode 属性中取错误码，由 ErrorMessage 属性中取错误信息。

## 7.2 证书解析方法

### ReadCertSN

**函数声明：** BSTR ReadCertSN(long nCertIndex);

**说明：**获取指定证书的序列号

参数:

- **nCertIndex**: 输入证书数组序号

**返回值:** 成功时返回证书序列号的十六进制编码,  
失败时返回空, 由 **ErrorCode** 属性中取错误码, 由 **ErrorMessage** 属性中取错误信息。

## ReadCertCN

**函数声明:** `BSTR ReadCertCN(long nCertIndex);`

**说明:** 获取指定证书的公共名称

参数:

- **nCertIndex**: 输入证书数组序号

**返回值:** 成功时返回证书的公共名称,  
失败时返回空, 由 **ErrorCode** 属性中取错误码, 由 **ErrorMessage** 属性中取错误信息。

## ReadCertDN

**函数声明:** `BSTR ReadCertDN(long nCertIndex);`

**说明:** 获取指定证书的主题

参数:

- **nCertIndex**: 输入证书数组序号

**返回值:** 成功时返回证书的主题,  
失败时返回空, 由 **ErrorCode** 属性中取错误码, 由 **ErrorMessage** 属性中取错误信息。

## ReadCertIssuer

**函数声明:** `BSTR ReadCertIssuer(long nCertIndex);`

**说明:** 获取指定证书的颁发者

参数:

- **nCertIndex**: 输入证书数组序号

**返回值:** 成功时返回证书的颁发者,  
失败时返回空, 由 **ErrorCode** 属性中取错误码, 由 **ErrorMessage** 属性中取错误信息。

## ReadCert

**函数声明:** `BSTR ReadCert(long nCertIndex);`

**说明:** 获取指定证书的内容(BASE64 编码)

参数:

- **nCertIndex**: 输入证书数组序号

**返回值:** 成功时返回证书内容的 BASE64 编码,  
失败时返回空, 由 ErrorCode 属性中取错误码, 由 ErrorMessage 属性中取错误信息。

## ReadCertBeginValidityTime

**函数声明:** BSTR ReadCertBeginValidityTime (long nCertIndex);

**说明:** 获取证书的起始有效期字符串(格式 YYYY-MM-DD hh:mm:ss)

**参数:**

- nCertIndex: 输入证书数组序号

**返回值:** 成功时返回证书起始有效期字符串,  
失败时返回空, 由 ErrorCode 属性中取错误码, 由 ErrorMessage 属性中取错误信息。

## ReadCertEndValidityTime

**函数声明:** BSTR ReadCertEndValidityTime (long nCertIndex);

**说明:** 获取证书的终止有效期字符串(格式 YYYY-MM-DD hh:mm:ss)

**参数:**

- nCertIndex: 输入证书数组序号

**返回值:** 成功时返回证书终止有效期字符串,  
失败时返回空, 由 ErrorCode 属性中取错误码, 由 ErrorMessage 属性中取错误信息。

## ReadCertExtension

**函数声明:** BSTR ReadCertExtension(BSTR szOID, long nCertIndex);

**说明:** 从选择的证书获得证书扩展信息

**参数:**

- szOID: 证书扩展域

可选值: 1.2.86.11.7.1 个人身份识别码  
1.2.86.11.7.2 个人社会保险号  
1.2.86.11.7.3 企业机构代码证号  
1.2.86.11.7.4 企业工商注册号  
1.2.86.11.7.5 企业(国税)税号  
1.2.86.77.1 企业(地税)税号

- nCertIndex: 输入证书数组序号

**返回值:** 成功时返回证书扩展域信息,  
失败时返回空, 由 ErrorCode 属性中取错误码, 由 ErrorMessage 属性中取错误信息。



## ReadCertExtensionCount

**函数声明：** long ReadCertExtensionCount( long nCertIndex);

**说明：** 从选择的证书获得证书扩展域总数

**参数：**

- **nCertIndex:** 输入证书数组序号

**返回值：** 成功时返回证书扩展域总数 $\geq 0$ ,

失败时返回 $< 0$ , 由 **ErrorCode** 属性中取错误码, 由 **ErrorMessage** 属性中取错误信息。

## ReadCertExtensionOID

**函数声明：** BSTR ReadCertExtensionOID( long nIndex);

**说明：** 从选择的证书获得证书扩展域包含的 **OID**,

前提条件: 已经成功地调用 **ReadCertExtensionCount()**函数

**参数：**

- **nIndex:** 输入证书扩展域序号(0--证书扩展域总数-1)

**返回值：** 成功时返回证书扩展域 **OID**,

失败时返回空, 由 **ErrorCode** 属性中取错误码, 由 **ErrorMessage** 属性中取错误信息。

## ReadCertExtensionValue

**函数声明：** BSTR ReadCertExtensionValue( BSTR szOID);

**说明：** 从选择的证书获得证书扩展域包含的值,

前提条件: 已经成功地调用 **ReadCertExtensionCount()**函数

**参数：**

- **szOID:** 输入证书扩展域 **OID**

可选值: 1.2.86.11.7.1 个人身份识别码

1.2.86.11.7.2 个人社会保险号

1.2.86.11.7.3 企业机构代码证号

1.2.86.11.7.4 企业工商注册号

1.2.86.11.7.5 企业(国税)税号

1.2.86.77.1 企业(地税)税号

**返回值：** 成功时返回返回证书扩展域值,

失败时返回空, 由 **ErrorCode** 属性中取错误码, 由 **ErrorMessage** 属性中取错误信息。

## ReadCertProviderName

**函数声明：** BSTR ReadCertProviderName (long nCertIndex);

**说明：** 获取指定证书提供者名称

**参数：**

- nCertIndex: 输入证书数组序号

**返回值：** 成功时返回证书提供者名称，

失败时返回空，由 ErrorCode 属性中取错误码，由 ErrorMessage 属性中取错误信息。

## CompareCertProviderName

**函数声明：** long CompareCertProviderName (long nCertIndex, BSTR CompareStr);

**说明：** 比较指定证书提供者名称

**参数：**

- nCertIndex: 输入证书数组序号
- CompareStr: 输入要比较的证书提供者名称

**返回值：** 一致时返回 0，不一致时返回>0，

调用失败时返回<0，由 ErrorCode 属性中取错误码，由 ErrorMessage 属性中取错误信息。

## 7.3 证书主题解析方法

### ReadCertDN\_OU

**函数声明：** BSTR ReadCertDN\_OU(long nCertIndex);

**说明：** 获取指定证书的主题 OU(组织单位)

**参数：**

- nCertIndex: 输入证书数组序号

**返回值：** 成功时返回证书的主题 OU(组织单位)，

失败时返回空，由 ErrorCode 属性中取错误码，由 ErrorMessage 属性中取错误信息。

### ReadCertDN\_OUa

**函数声明：** BSTR ReadCertDN\_OUa(long nCertIndex);

**说明：** 获取指定证书的主题 OU@(企业代码证号)

**参数：**

- nCertIndex: 输入证书数组序号

**返回值：**成功时返回指定证书的主题 OU@(企业代码证号)，  
失败时返回空，由 ErrorCode 属性中取错误码，由 ErrorMessage 属性中取错误信息。

## ReadCertDN\_OUi

**函数声明：** BSTR ReadCertDN\_OUi(long nCertIndex);

**说明：**获取指定证书的主题 OU!(备注项)

**参数：**

- nCertIndex: 输入证书数组序号

**返回值：**成功时返回指定证书的主题 OU!(备注项)，  
失败时返回空，由 ErrorCode 属性中取错误码，由 ErrorMessage 属性中取错误信息。

## ReadCertDN\_O

**函数声明：** BSTR ReadCertDN\_O(long nCertIndex);

**说明：**获取指定证书的主题 O(组织)

**参数：**

- nCertIndex: 输入证书数组序号

**返回值：**成功时返回指定证书的主题 O(组织)，  
失败时返回空，由 ErrorCode 属性中取错误码，由 ErrorMessage 属性中取错误信息。

## ReadCertDN\_L

**函数声明：** BSTR ReadCertDN\_L(long nCertIndex);

**说明：**获取指定证书的主题 L(市)

**参数：**

- nCertIndex: 输入证书数组序号

**返回值：**成功时返回指定证书的主题 L(市)，  
失败时返回空，由 ErrorCode 属性中取错误码，由 ErrorMessage 属性中取错误信息。

## ReadCertDN\_S

**函数声明：** BSTR ReadCertDN\_S(long nCertIndex);

**说明：**获取指定证书的主题 S(省)

**参数：**

- nCertIndex: 输入证书数组序号

**返回值：**成功时返回指定证书的主题 S(省)，

失败时返回空, 由 `ErrorCode` 属性中取错误码, 由 `ErrorMessage` 属性中取错误信息。

## ReadCertDN\_C

**函数声明:** `BSTR ReadCertDN_C(long nCertIndex);`

**说明:** 获取指定证书的主题 C(国家)

**参数:**

- `nCertIndex`: 输入证书数组序号

**返回值:** 成功时返回指定证书的主题 C(国家),

失败时返回空, 由 `ErrorCode` 属性中取错误码, 由 `ErrorMessage` 属性中取错误信息。

## ReadCertDN\_E

**函数声明:** `BSTR ReadCertDN_E(long nCertIndex);`

**说明:** 获取指定证书的主题 E(Email)

**参数:**

- `nCertIndex`: 输入证书数组序号

**返回值:** 成功时返回指定证书的主题 E(Email),

失败时返回空, 由 `ErrorCode` 属性中取错误码, 由 `ErrorMessage` 属性中取错误信息。

## 7.4 证书查看

### ViewCertWnd

**函数声明:** `long ViewCertWnd(long Index);`

**说明:** 显示选择后的证书窗口, 前提条件: 已经成功初始化证书。

**参数:**

- `Index`: 证书索引号

**返回值:** 成功返回 0,

失败时返回<0, 由 `ErrorCode` 属性中取错误码, 由 `ErrorMessage` 属性中取错误信息。

## 7.5 CRL 操作

### InitCRL

**函数声明:** `long InitCRL (BSTR CRLBase64Data);`

**说明：**通过数据(BASE64 编码)初始化 CRL。

**参数：**

- CRLBase64Data: CRL 数据(BASE64 编码)

**返回值：**成功时返回 0，

失败时返回<0，由 ErrorCode 属性中取错误码，由 ErrorMessage 属性中取错误信息。

提示：还可以通过从辽宁 CA 的 LDAP 服务器来初始化 CRL。参见 ReadCRLFromLdap。

## VerifyCRL

**函数声明：**long VerifyCRL (BSTR CRLBase64Data, long IsVerifyTime, BSTR sSystemTime);

**说明：**验证 CRL 有效性。如果验证成功，初始化 CRL 对象，CRL 数据可以从 LDAP 服务器下载。

**参数：**

- CRLBase64Data: CRL 数据(BASE64 编码)
- IsVerifyTime: 是否校验 CRL 时间有效性
  - 0 = 不校验 CRL 时间
  - 1 = 校验 CRL 时间，并且下一个参数有效
- sSystemTime: 外部输入要验证的日期和时间  
输入格式必须为: yyyy.mm.dd hh:mm:ss 或 yyyy.m.d. h:m:s，其中日期与时间之间用空格隔开，如果为"" = 取本机系统时间

**返回值：**有效返回 0，否则返回-1: 未启用无效，1: 超期无效

失败时返回其它值，由 ErrorCode 属性中取错误码，由 ErrorMessage 属性中取错误信息。

## CRLVerifyCert

**函数声明：**long CRLVerifyCert (long nCertIndex);

**说明：**使用 CRL 校证书，前提条件：已经成功地初始化 CRL 对象。

**参数：**

- Index: 证书索引号

**返回值：**证书有效返回 0，

失败时返回<0，由 ErrorCode 属性中取错误码，由 ErrorMessage 属性中取错误信息。

## 7.6 哈希操作方法

### HashData

**函数声明:** BSTR HashData(BSTR sSourceData, BSTR HashAlgo);

**说明:** 哈希数据操作

**参数:**

- sSourceData: 原文数据
  - HashAlgo: 哈希算法
- |                                      |                     |
|--------------------------------------|---------------------|
| szOID_OIWSEC_sha1 = "1.3.14.3.2.26"  | 产生 28 字节(BASE64 编码) |
| szOID_RSA_MD5 = "1.2.840.113549.2.5" | 产生 24 字节(BASE64 编码) |
| szOID_RSA_MD4 = "1.2.840.113549.2.4" | 产生 24 字节(BASE64 编码) |
| szOID_RSA_MD2 = "1.2.840.113549.2.2" | 产生 24 字节(BASE64 编码) |

**返回值:** 成功时返回 BASE64 编码哈希值,

失败时返回空, 由 ErrorCode 属性中取错误码, 由 ErrorMessage 属性中取错误信息。

### VerifyHashData

**函数声明:** long VerifyHashData (BSTR sSourceData, BSTR sHashData, BSTR HashAlgo);

**说明:** 验证哈希数据操作

**参数:**

- sSourceData: 原文数据
  - sHashData: 原 BASE64 编码哈希值
  - HashAlgo: 哈希算法
- :
- |                                      |
|--------------------------------------|
| szOID_OIWSEC_sha1 = "1.3.14.3.2.26"  |
| szOID_RSA_MD5 = "1.2.840.113549.2.5" |
| szOID_RSA_MD4 = "1.2.840.113549.2.4" |
| szOID_RSA_MD2 = "1.2.840.113549.2.2" |

**返回值:** 成功时返回 0,

失败时返回<0, 由 ErrorCode 属性中取错误码, 由 ErrorMessage 属性中取错误信息。

## 7.7 签名操作方法

### SignHash

**函数声明:** BSTR SignHash (BSTR sSourceData, BSTR HashAlgo, BSTR pPin);

**说明:** 进行签名操作 (使用客户端证书)

**参数:**

- sSourceData: 原文数据
- HashAlgo: 哈希算法

szOID\_OIWSEC\_sha1 = "1.3.14.3.2.26"      产生 176 字节(BASE64 编码)

szOID\_RSA\_MD5 = "1.2.840.113549.2.5"      产生 176 字节(BASE64 编码)

szOID\_RSA\_MD2 = "1.2.840.113549.2.2"      产生 176 字节(BASE64 编码)

- pPin: 用户 Key 口令

如果输入正确的口令, 则不弹出输入口令窗口, 直接签名哈希数据。

如果输入错误的口令, 则弹出输入口令窗口

**返回值:** 成功时返回 BASE64 编码签名哈希值,

失败时返回空, 由 ErrorCode 属性中取错误码, 由 ErrorMessage 属性中取错误信息。

## VerifySignHash

**函数声明:** long VerifySignHash (BSTR sSourceData, BSTR sHashData, BSTR HashAlgo, long nCertIndex);

**说明:** 验证签名操作

**参数:**

- sSourceData: 原文数据
- sHashData: 原 BASE64 编码哈希签名值
- HashAlgo: 哈希算法
  - szOID\_OIWSEC\_sha1 = "1.3.14.3.2.26"
  - szOID\_RSA\_MD5 = "1.2.840.113549.2.5"
  - szOID\_RSA\_MD2 = "1.2.840.113549.2.2"
- nCertIndex: 指定验证用的公钥证书

**返回值:** 成功时返回 0,

失败时返回<0, 由 ErrorCode 属性中取错误码, 由 ErrorMessage 属性中取错误信息。

## 7.8 PKCS7 签名数据操作方法

### SignDataEx

**函数声明:** BSTR SignDataEx (BSTR szSrc,  
                                   BSTR sSignAlgo,  
                                   long IsAddSignCert,  
                                   long IsAddSrcData,  
                                   BSTR szInnerOid,  
                                   long IsAddTime,  
                                   BSTR pPin);

**说明:** 进行签名数据操作(使用客户端证书)。

**参数:**

- szSrc : 原文数据
- sSignAlgo: 指定签名算法

szOID\_OIWSEC\_sha1 = "1.3.14.3.2.26"

szOID\_RSA\_MD5 = "1.2.840.113549.2.5"

szOID\_RSA\_MD2 = "1.2.840.113549.2.2"

- IsAddSignCert 是否在结果中携带证书  
0 = 不携带证书  
1 = 携带证书
- IsAddSrcData 是否在结果中携带原文  
0 = 不携带原文  
1 = 携带原文
- szInnerOid: 数据类型 OID, (默认:NULL)  
szOID\_TSP\_TSTInfo = "1.2.840.113549.1.9.16.1.4"
- IsAddTime: 是否添加签名时间  
0: 不进行时间编码  
1: 取当前系统时间, 进行时间编码  
2: 从时间戳服务器取得时间, 必须首先设置时间戳服务器 URL。请参考 7.14 章节的时间戳操作。

- pPin: 用户 Key 口令

如果输入正确的口令, 则不弹出输入口令窗口, 直接签名数据。

如果输入错误的口令, 则弹出输入口令窗口

**返回值:** 成功时返回签名数据(BASE64 编码),

失败时返回空, 由 ErrorCode 属性中取错误码, 由 ErrorMessage 属性中取错误信息。

## SignData //吉大兼容

**函数声明:** BSTR SignData(BSTR szSrc);

**说明:** 进行签名数据操作 (携带证书, 携带原文) (使用客户端证书)。

证书如有私钥保护口令, 弹出输入口令窗口, 手工输入私钥保护口令

**参数:**

- szSrc : 原文数据

其他默认参数: 签名算法: szOID\_RSA\_MD5 = "1.2.840.113549.2.5",

携带证书,

携带原文,

数据类型 OID=NULL,

是否附加时间编码: 0: 不进行时间编码

**返回值:** 成功时返回签名数据(BASE64 编码),

失败时返回空, 由 ErrorCode 属性中取错误码, 由 ErrorMessage 属性中取错误信息。

## VerifySignEx

**函数声明:** BSTR VerifySignEx (BSTR szSrc, BSTR szSign, long nCertIndex);

**说明:** 验证签名数据操作。验证由 SignData 或者 SignDataEx 生成的签名数据。



**参数:**

- szSrc: 原文数据(如果为空, 表示此签名数据中携带原文, 否则输入原文数据)
- szSign: 签名数据(BASE64 编码)
- long nCertIndex: 指定使用的验证签名公钥证书
  - 0 = 自己的签名公钥证书
  - 2 = 服务器的公钥证书
  - 4 = 第三方的公钥证书
  - 1 = 一个特例, 即不用指定证书, 签名数据内包含签名者证书

**返回值:** 成功时返回原文,

失败时返回空, 由 ErrorCode 属性中取错误码, 由 ErrorMessage 属性中取错误信息。

## VerifySign //吉大兼容

**函数声明:** long VerifySign(BSTR szSrc, BSTR szSign);

**说明:** 验证签名数据操作。此签名数据中携带证书, 验证由 SignData 或者 SignDataEx 生成的签名数据。

**注意:** 不能验证签名数据中不携带证书, 进行时间编码, 并且不能返回原文。

**参数:**

- szSrc: 原文数据(如果为空, 表示此签名数据中携带原文, 否则输入原文数据)
- szSign: 签名数据(BASE64 编码)

**返回值:** 成功时返回 0,

失败时返回<0, 由 ErrorCode 属性中取错误码, 由 ErrorMessage 属性中取错误信息。

## SignFileEx

**函数声明:** BSTR SignFileEx (BSTR szSrcFile, BSTR szEvFile,  
BSTR sSignAlgo, long IsAddSignCert,  
long IsAddTime, BSTR pPin);

**说明:** 进行签名数据文件操作, 使用客户端证书, 产生包含原文、二进制编码的签名数据文件。

**参数:**

- szSrcFile : 输入原文数据文件名
- szEvFile : 输出签名数据文件名(BIN 编码)
- sSignAlgo: 指定签名算法
  - szOID\_OIWSEC\_sha1 = "1.3.14.3.2.26"
  - szOID\_RSA\_MD5 = "1.2.840.113549.2.5"
  - szOID\_RSA\_MD2 = "1.2.840.113549.2.2"
- IsAddSignCert 是否在结果中携带证书
  - 0 = 不携带证书
  - 1 = 携带证书
- IsAddTime:是否添加签名时间
  - 0: 不进行时间编码

1: 取当前系统时间, 进行时间编码

2、从时间戳服务器取得时间, 必须首先设置时间戳服务器 URL。请参考 7.14 章节的时间戳操作。

● **pPin:** 用户 Key 口令

如果输入正确的口令, 则不弹出输入口令窗口, 直接签名数据。

如果输入错误的口令, 则弹出输入口令窗口

**返回值:** 成功是分两种情况,

如果 szEvpFile 参数为空, 则返回签名的 BASE64 编码数据,

如果 szEvpFile 参数不为空, 则返回 szEvpFile 名称

失败时返回空, 由 ErrorCode 属性中取错误码, 由 ErrorMessage 中取错误信息。

## VerifySignFileEx

**函数声明:** BSTR VerifySignFileEx (BSTR szEvpFile, BSTR szSrcFile, long nSignCertIndex);

**说明:** 验证签名数据文件操作, 验证由 SignFileEx 生成的包含原文、二进制编码签名数据文件。

**参数:**

● **szEvpFile:** 输入签名数据文件名(BIN 编码)

● **szSrcFile:** 输出原文数据的文件名

● **long nCertIndex:** 指定使用的验证签名公钥证书

0 = 自己的签名公钥证书

2 = 服务器的公钥证书

4 = 第三方的公钥证书

-1 = 一个特例, 即不用指定证书, 签名数据内包含签名者证书

**返回值:** 成功是分两种情况,

如果 szSrcFile 参数为空, 则返回原文数据的 BASE64 编码数据,

如果 szSrcFile 参数不为空, 则返回 szSrcFile 名称

失败时返回空, 由 ErrorCode 属性中取错误码, 由 ErrorMessage 中取错误信息。

## ReturnSignerCertData

**函数声明:** BSTR ReturnSignerCertData ();

**说明:** 返回验证签名数据后的公钥证书数据, 前提条件: 已经成功地验证签名数据。

**注意:** 如果签名数据中不包含签名者证书, 则无法返回证书数据。

**参数:**

**返回值:** 成功时返回 BASE64 编码证书数据,

失败时返回空, 由 ErrorCode 属性中取错误码, 由 ErrorMessage 属性中取错误信息。

## ReturnSignerCertSN

**函数声明:** BSTR ReturnSignerCertSN (BSTR pbBase64SignedMessage);

说明：返回签名数据后的签名者证书序列号。

注意：如果签名数据中不包含证书，则无法返回证书序列号。

参数：

- pbBase64SignedMessage: BASE64 编码签名数据

返回值：成功时返回签名者证书序列号，

失败时返回空，由 ErrorCode 属性中取错误码，由 ErrorMessage 属性中取错误信息。

## ReturnSigningTime

函数声明：BSTR ReturnSigningTime (BSTR pbBase64SignedMessage);

说明：返回签名数据后的签名时间。(格式 YYYY-MM-DD hh:mm:ss)。

注意：如果签名数据时不包含签名时间，则无法返回签名时间。

参数：

- pbBase64SignedMessage: BASE64 编码签名数据

返回值：成功时返回签名时间字符串，

失败时返回空，由 ErrorCode 属性中取错误码，由 ErrorMessage 属性中取错误信息。

## 7.9 PKCS7 加密信封操作方法

### EncryptDataEx

函数声明：BSTR EncryptDataEx (BSTR szSrc, long EncType , BSTR EncryptAlgo);

说明：进行加密信封操作。

参数：

- szSrc: 原文数据
- EncType: 指定使用的公钥证书
  - 1 = 自己的公钥证书
  - 2 = 服务器的公钥证书
  - 3 = 第三方的公钥证书
  - 4 = 同时使用服务器的公钥证书和第三方的公钥证书
- EncryptAlgo: 加密算法
  - szOID\_RSA\_DES\_EDE3\_CBC = "1.2.840.113549.3.7"
  - szOID\_OIWSEC\_desCBC = "1.3.14.3.2.7"
  - szOID\_RSA\_RC4 = "1.2.840.113549.3.4"
  - szOID\_RSA\_RC2CBC = "1.2.840.113549.3.2"

返回值：成功时返回加密信封数据(BASE64 编码)，

失败时返回空，由 ErrorCode 属性中取错误码，由 ErrorMessage 属性中取错误信息。

## EncryptData //吉大兼容

**函数声明:** BSTR EncryptData(BSTR szSrc, long EncType);

**说明:** 进行加密信封操作。

**参数:**

- szSrc: 原文数据
- EncType: 指定使用的公钥证书
  - 1 = 自己的公钥证书
  - 2 = 服务器的公钥证书
  - 3 = 第三方的公钥证书
  - 4 = 同时使用服务器的公钥证书和第三方的公钥证书

**默认加密算法:** szOID\_RSA\_DES\_EDE3\_CBC = "1.2.840.113549.3.7"

**返回值:** 成功时返回加密信封数据(BASE64 编码),

失败时返回空, 由 ErrorCode 属性中取错误码, 由 ErrorMessage 属性中取错误信息。

## DecryptData //吉大兼容

**函数声明:** BSTR DecryptData(BSTR szEnc);

**说明:** 进行加密信封的解密操作。

**参数:**

- szEnc: 加密信封(BASE64 编码)

**返回值:** 成功时返回解密原文,

失败时返回空, 由 ErrorCode 属性中取错误码, 由 ErrorMessage 属性中取错误信息。

## EncryptFileEx

**函数声明:** BSTR EncryptFileEx (BSTR szSrcFile, BSTR szEncFile, long EncType , BSTR EncryptAlgo);

**说明:** 进行加密信封操作。产生二进制编码的密文信封数据。

**参数:**

- szSrcFile: 输入原文数据文件名全路径
- szEncFile: 输出密文数据文件名全路径(BIN 编码)
- EncType: 指定使用的公钥证书
  - 1 = 自己的公钥证书
  - 2 = 服务器的公钥证书
  - 3 = 第三方的公钥证书
  - 4 = 同时使用服务器的公钥证书和第三方的公钥证书
- EncryptAlgo: 加密算法
  - szOID\_RSA\_DES\_EDE3\_CBC = "1.2.840.113549.3.7"
  - szOID\_OIWSEC\_desCBC = "1.3.14.3.2.7"
  - szOID\_RSA\_RC4 = "1.2.840.113549.3.4"
  - szOID\_RSA\_RC2CBC = "1.2.840.113549.3.2"

**返回值：**成功是分两种情况，

如果 szEncFile 参数为空，则返回密文的 BASE64 编码数据，

如果 szEncFile 参数不为空，则返回 szEncFile 名称

失败时返回空，由 ErrorCode 属性中取错误码，由 ErrorMessage 中取错误信息。

## DecryptFile

**函数声明：** BSTR DecryptFile(BSTR szEncFile, BSTR szSrcFile);

**说明：**进行加密信封的解密操作。

**参数：**

- szEncFile: 输入加密信封文件名全路径(BIN 编码)
- szSrcFile: 输出解密后的原文文件名全路径

**返回值：**成功是分两种情况，

如果 szSrcFile 参数为空，则返回解密后的原文 BASE64 编码数据，

如果 szSrcFile 参数不为空，则返回 szSrcFile 文件名称

失败时返回空，由 ErrorCode 属性中取错误码，由 ErrorMessage 中取错误信息。

## 7.10 PKCS7 签名加密信封操作方法

### SignEnvDataEx

**函数声明：** BSTR SignEnvDataEx (BSTR szSrc,  
long nEncCertType,  
BSTR SignAlgo,  
BSTR EncryptAlgo,  
long IsAddSignCert,  
long IsAddTime,  
BSTR pPin);

**说明：**进行签名加密信封操作，默认携带原文数据。

**参数：**

- szSrc: 待签名加密的数据
- nEncCertType: 指定公钥证书
  - 1 = 客户加密证书
  - 2 = 服务器加密证书
  - 3 = 第三方加密证书
  - 4 = 同时使用服务器的公钥证书和第三方的公钥证书
- SignAlgo: 签名算法
  - szOID\_OIWSEC\_sha1 = "1.3.14.3.2.26"
  - szOID\_RSA\_MD5 = "1.2.840.113549.2.5"

szOID\_RSA\_MD2 = "1.2.840.113549.2.2"

- **EncryptAlgo:** 加密算法

szOID\_RSA\_DES\_EDE3\_CBC = "1.2.840.113549.3.7"

szOID\_OIWSEC\_desCBC = "1.3.14.3.2.7"

szOID\_RSA\_RC4 = "1.2.840.113549.3.4"

szOID\_RSA\_RC2CBC = "1.2.840.113549.3.2"

- **IsAddSignCert:** 是否在结果中携带证书

0 = 不携带证书

1 = 携带证书

- **IsAddTime:** 是否添加签名时间

0 = 不进行时间编码

1 = 取当前系统时间, 进行时间编码

2、从时间戳服务器取得时间, 必须首先设置时间戳服务器 URL。请参考 7.14

章节的时间戳操作。

- **pPin:** 用户 Key 口令

如果输入正确的口令, 则不弹出输入口令窗口, 直接哈希签名数据。

如果输入错误的口令, 则弹出输入口令窗口

**返回值:** 成功时返回签名加密信封(BASE64 编码),

失败时返回空, 由 **ErrorCode** 属性中取错误码, 由 **ErrorMessage** 属性中取错误信息。

## SignEnvData //吉大兼容

**函数声明:** BSTR SignEnvData(BSTR szSrc);

**说明:** 进行签名加密信封操作, 默认携带原文数据。

**参数:**

- **szSrc:** 待签名加密的数据

默认加密证书: 使用服务器加密证书 = 2

默认算法:

签名算法: szOID\_RSA\_MD5RSA = "1.2.840.113549.1.1.4"

加密算法: szOID\_RSA\_DES\_EDE3\_CBC = "1.2.840.113549.3.7"

默认是否携带证书: 1 = 携带证书

默认是否添加签名时间: 0 = 不进行时间编码

默认用户 Pin 码: 为空

**返回值:** 成功时返回签名加密信封(BASE64 编码),

失败时返回空, 由 **ErrorCode** 属性中取错误码, 由 **ErrorMessage** 属性中取错误信息。

## VfyDecDataEx

**函数声明:** BSTR VfyDecDataEx(BSTR szEnc, long nSignCertIndex);

**说明:** 验证解密由 SignEncData 或者 SignEncDataEx 产生的签名加密信封。

**参数:**

- **szEnc:** 签名加密信封(BASE64 编码)
- **nSignCertIndex:** 指定验证签名证书在证书数组中的索引(从 0 开始)
  - 0 = 自己的签名公钥证书
  - 2 = 服务器的签名公钥证书
  - 4 = 第三方的签名公钥证书
  - 1 = 一个特例, 即不用指定证书, 签名数据内包含签名者证书

**返回值:** 成功时返回原文数据,

失败时返回空, 由 **ErrorCode** 属性中取错误码, 由 **ErrorMessage** 属性中取错误信息。

## VfyDecData //吉大兼容

**函数声明:** BSTR VfyDecData(BSTR szEnc);

**说明:** 验证解密由 **SignEncData** 或者 **SignEncDataEx** 产生的签名加密信封。

**注意:** 不能验证不携带签名证书, 进行时间编码的签名加密信封。

**参数:**

- **szEnc:** 签名加密信封(BASE64 编码)

**返回值:** 成功时返回验证解密后的原文数据,

失败时返回空, 由 **ErrorCode** 属性中取错误码, 由 **ErrorMessage** 属性中取错误信息。

## SignEncFileEx

**函数声明:** BSTR SignEncFileEx (BSTR szSrcFile,  
                                   BSTR szEvpFile,  
                                   long nEncCertType,  
                                   BSTR SignAlgo,  
                                   BSTR EncryptAlgo,  
                                   long IsAddSignCert,  
                                   long IsAddTime,  
                                   BSTR pPin);

**说明:** 对文件进行签名加密信封操作, 返回携带原文的二进制编码信封数据文件。

**参数:**

- **szSrcFile:** 输入原文文件名(全路径)
- **szEvpFile:** 输出的信封文件名(全路径)
- **nEncCertType:** 指定公钥证书
  - 1 = 客户加密证书
  - 2 = 服务器加密证书
  - 3 = 第三方加密证书
  - 4 = 同时使用服务器的公钥证书和第三方的公钥证书
- **SignAlgo:** 签名算法
  - szOID\_OIWSEC\_sha1 = "1.3.14.3.2.26"
  - szOID\_RSA\_MD5 = "1.2.840.113549.2.5"
  - szOID\_RSA\_MD2 = "1.2.840.113549.2.2"



- **EncryptAlgo:** 加密算法
  - szOID\_RSA\_DES\_EDE3\_CBC = "1.2.840.113549.3.7"
  - szOID\_OIWSEC\_desCBC = "1.3.14.3.2.7"
  - szOID\_RSA\_RC4 = "1.2.840.113549.3.4"
  - szOID\_RSA\_RC2CBC = "1.2.840.113549.3.2"
- **IsAddSignCert:** 是否在结果中携带证书
  - 0 = 不携带证书
  - 1 = 携带证书
- **IsAddTime:** 是否添加签名时间
  - 0 = 不进行时间编码
  - 1 = 取当前系统时间, 进行时间编码
  - 2 = 从时间戳服务器取得时间, 必须首先设置时间戳服务器 URL。请参考 7.14 章节的时间戳操作。
- **pPin:** 用户 Key 口令
 

如果输入正确的口令, 则不弹出输入口令窗口, 直接签名数据。

如果输入错误的口令, 则弹出输入口令窗口

**返回值:** 成功是分两种情况,

如果 szEvpFile 参数为空, 则返回签名加密信封的 BASE64 编码数据,

如果 szEvpFile 参数不为空, 则返回 szEvpFile 名称

失败时返回空, 由 ErrorCode 属性中取错误码, 由 ErrorMessage 中取错误信息。

## SignEncFile //吉大兼容

**函数声明:** BSTR SignEncFile(BSTR szSrcFile, BSTR szEvpFile);

**说明:** 对文件进行签名加密信封操作, 返回携带原文的二进制编码信封数据文件。

**参数:**

- **szSrcFile:** 输入原文文件名(全路径)
- **szEvpFile:** 输出的信封文件名(全路径)

默认加密证书: 使用服务器加密证书 = 2

默认签名算法: szOID\_RSA\_MD5 = "1.2.840.113549.2.5"

默认加密算法: szOID\_RSA\_DES\_EDE3\_CBC = "1.2.840.113549.3.7"

默认是否携带证书: 1 = 携带证书

默认是否添加签名时间: 0 = 不进行时间编码

默认用户 Pin 码: 为空

**返回值:** 成功是分两种情况,

如果 szEvpFile 参数为空, 则返回签名加密信封的 BASE64 编码数据,

如果 szEvpFile 参数不为空, 则返回 szEvpFile 名称

失败时返回空, 由 ErrorCode 属性中取错误码, 由 ErrorMessage 中取错误信息。



## VfyDecFileEx

**函数声明:** BSTR VfyDecFileEx(BSTR szEvpFile, BSTR szSrcFile, long nSignCertIndex);

**说明:** 验证解密由 SignEncFile 或者 SignEncFileEx 产生的签名加密信封。

**参数:**

- szEvpFile: 输入信封文件名全路径(二进制编码)
- szSrcFile: 输出原文文件名全路径
- nSignCertIndex: 指定验证签名证书在证书数组中的索引(从 0 开始)
  - 0 = 自己的签名公钥证书
  - 2 = 服务器的签名公钥证书
  - 4 = 第三方的签名公钥证书
  - 1 = 一个特例, 即不用指定证书, 签名数据内包含签名者证书

**返回值:** 成功是分两种情况,

如果 szSrcFile 参数为空, 则返回原文数据的 BASE64 编码数据,

如果 szEvpFile 参数不为空, 则返回 szSrcFile 名称

失败时返回空, 由 ErrorCode 属性中取错误码, 由 ErrorMessage 中取错误信息。

## VfyDecFile

## //吉大兼容

**函数声明:** BSTR VfyDecFile(BSTR szEvpFile, BSTR szSrcFile);

**说明:** 对签名加密信封文件进行解密验证操作, 验证解密由 SignEncFile 产生的签名加密信封文件。

**注意:** 不能验证不携带签名证书或者进行时间编码的签名加密信封。

**参数:**

- szEvpFile: 输入信封文件名全路径(二进制编码)
- szSrcFile: 输出原文文件名全路径

**返回值:** 成功是分两种情况,

如果 szSrcFile 参数为空, 则返回原文数据的 BASE64 编码数据,

如果 szEvpFile 参数不为空, 则返回 szSrcFile 名称

失败时返回空, 由 ErrorCode 属性中取错误码, 由 ErrorMessage 中取错误信息。

## 7.11 对称加解密操作方法

### SymCipherInit

**函数声明:** long SymCipherInit(BSTR CipherAlgo, BSTR pbKey);

**说明:** 对称加解密初始化, 必须进行此初始化后, 才可以进行对称加解密操作。必须输入口令。

**参数:**

- **CipherAlgo:** 加解密算法  
3DES 算法: "1.2.840.113549.3.7" = szOID\_RSA\_DES\_EDE3\_CBC  
DES 算法: "1.3.14.3.2.7" = szOID\_OIWSEC\_desCBC,  
RC4 算法: "1.2.840.113549.3.4" = szOID\_RSA\_RC4,  
RC2 算法: "1.2.840.113549.3.2" = szOID\_RSA\_RC2CBC
  - **pbKey:** 密钥口令(4-8) , 必须输入口令
- 返回值:** 成功时返回 0,  
失败时返回<0, 由 **ErrorCode** 属性中取错误码, 由 **ErrorMessage** 属性中取错误信息。

## SymEncryptData

**函数声明:** BSTR SymEncryptData(BSTR SourceData);

**说明:** 对称加密数据, 必须调用 SymCipherInit()初始化后, 才可以进行对称加密操作。

**参数:**

- **SourceData:** 原文数据

**返回值:** 成功时返回 BASE64 编码密文数据,

失败时返回空, 由 **ErrorCode** 属性中取错误码, 由 **ErrorMessage** 属性中取错误信息。

## SymDecryptData

**函数声明:** BSTR SymDecryptData(BSTR CryptoData);

**说明:** 对称解密数据, 必须调用 SymCipherInit()初始化后, 才可以进行对称解密操作。

**参数:**

- **CryptoData:** BASE64 编码密文数据

**返回值:** 成功时返回解密后的原文数据,

失败时返回空, 由 **ErrorCode** 属性中取错误码, 由 **ErrorMessage** 属性中取错误信息。

## 7.12 LDAP 相关操作方法

### ReadRootCertFromLdap

**函数声明:** BSTR ReadRootCertFromLdap (BSTR szLDAPServerAddress, long dwPort);

**说明:** 从 LDAP 服务器获得根证书(BASE64 编码), 同时初始化根证书对象, 并清空证书数组中的其他对象。

**注:** 每次连接 LDAP 服务器, 需要 5 秒时间, 请耐心等待。

**参数:**

- **szLDAPServerAddress:** LDAP 服务器地址, 如果输入 NULL = 默认从 ldap.ln-ca.com 下载
- **dwPort:** LDAP 服务器端口, 如果输入 0 = 默认从 391 下载

**返回值:** 成功时返回 BASE64 编码公钥根证书数据,  
失败时返回空, 由 ErrorCode 属性中取错误码, 由 ErrorMessage 属性中取错误信息。

## ReadCRLFromLdap

**函数声明:** BSTR ReadCRLFromLdap (BSTR szLDAPServerAddress, long dwPort);

**说明:** 从 LDAP 服务器获得 CRL(BASE64 编码), 同时初始化 CRL 对象。

**注:** 每次连接 LDAP 服务器, 需要 5 秒时间, 请耐心等待。

**参数:**

- szLDAPServerAddress: LDAP 服务器地址, 如果输入 NULL = 默认从 ldap.ln-ca.com 下载
- dwPort: LDAP 服务器端口, 如果输入 0 = 默认从 391 下载

**返回值:** 成功时返回 BASE64 编码 CRL 数据,  
失败时返回空, 由 ErrorCode 属性中取错误码, 由 ErrorMessage 属性中取错误信息。

## ReadCertFromLdap

**函数声明:** BSTR ReadCertFromLdap (BSTR szCertSerialNumber,  
BSTR szLDAPServerAddress,  
long dwPort);

**说明:** 从 LDAP 服务器获得指定证书序列号的公钥证书(BASE64 编码)。

**注:** 每次连接 LDAP 服务器, 需要 5 秒时间, 请耐心等待。

**参数:**

- szCertSerialNumber: 证书序列号
- szLDAPServerAddress: LDAP 服务器地址, 如果输入 NULL = 默认从 ldap.ln-ca.com 下载
- dwPort: LDAP 服务器端口, 如果输入 0 = 默认从 391 下载

**返回值:** 成功时返回 BASE64 编码公钥证书数据,  
失败时返回空, 由 ErrorCode 属性中取错误码, 由 ErrorMessage 属性中取错误信息。

## ReturnCertType

**函数声明:** long ReturnCertType ();

**说明:** 从 LDAP 获得查询证书后, 返回证书类型号。

**注:** 前提条件是已经成功地从 LDAP 服务器上查询到证书。

目前设定的证书种类如下:

- 4: 基本 CA 单证书
- 5: 基本 CA 双证书
- 102: 个人身份证书

- 103: 个人代码签名证书
- 104: 单位身份证书
- 106: 个人电子商务证书
- 107: 单位电子商务证书
- 108: 设备证书
- 109: CRL 验证个人身份证书
- 110: 辽宁个人身份证书
- 111: 辽宁个人代码签名证书
- 112: 辽宁个人电子商务证书
- 113: 辽宁 CRL 验证个人身份证书
- 114: 辽宁单位身份证书
- 115: 辽宁单位代码签名证书
- 116: 辽宁单位电子商务证书
- 117: 辽宁设备证书
- 118: 单位代码签名证书
- 127: 国家根 CA 测试

参数:

返回值: 成功时返回证书类型号,

失败时返回<0, 由 ErrorCode 属性中取错误码, 由 ErrorMessage 属性中取错误信息。

## ReturnCertStatus

函数声明: long ReturnCertStatus ();

说明: 从 LDAP 获得查询证书后, 返回证书状态。

注: 前提条件是已经成功地从 LDAP 服务器上查询证书。

目前设定的证书状态如下:

- 3: 新申请 (证书未下载)
- 5: 证书使用中
- 6: 证书注销

参数:

返回值: 成功时返回证书状态,

失败时返回<0, 由 ErrorCode 属性中取错误码, 由 ErrorMessage 属性中取错误信息。

## ReturnCertTypeStr

函数声明: BSTR ReturnCertTypeStr (long CertType);

说明: 证书类型码转换为字符串。

目前设定的证书种类如下:

- 4: 基本 CA 单证书
- 5: 基本 CA 双证书
- 102: 个人身份证书
- 103: 个人代码签名证书
- 104: 单位身份证书

- 106: 个人电子商务证书
- 107: 单位电子商务证书
- 108: 设备证书
- 109: CRL 验证个人身份证书
- 110: 辽宁个人身份证书
- 111: 辽宁个人代码签名证书
- 112: 辽宁个人电子商务证书
- 113: 辽宁 CRL 验证个人身份证书
- 114: 辽宁单位身份证书
- 115: 辽宁单位代码签名证书
- 116: 辽宁单位电子商务证书
- 117: 辽宁设备证书
- 118: 单位代码签名证书
- 127: 国家根 CA 测试

参数:

- CertType: 证书类型码

返回值: 成功时返回证书类型字符串, 否则返回空。

## ReturnCertStatusStr

函数声明: BSTR ReturnCertStatusStr (long CertStatus);

说明: 证书状态编码转换为字符串。

目前设定的证书状态如下:

- 3: 新申请 (证书未下载)
- 5: 证书使用中
- 6: 证书注销

参数:

- CertStatus: 证书状态编码

返回值: 成功时返回证书状态字符串, 否则返回空。

## 7.13 时间戳操作方法

### SetTimeStampServerUrl

函数声明: long SetTimeStampServerUrl(BSTR sUrl);

说明: 设置时间戳服务器地址。

参数:

- sUrl: 时间戳服务器地址

返回值: 成功时返回时间戳服务器地址总数。

## AppendTimeStampServerUrl

**函数声明:** long AppendTimeStampServerUrl(BSTR sUrl);

**说明:** 追加时间戳服务器地址。

**参数:**

- sUrl: 时间戳服务器地址

**返回值:** 成功时返回时间戳服务器地址总数。

## RequestTimeStamp

**函数声明:** long RequestTimeStamp (BSTR sData, BSTR szHashAlgorithmOID);

**说明:** 发送时间戳请求。前提条件: 已经设置或追加时间戳服务器地址

**参数:**

- sData: 请求原文数据
- szHashAlgorithmOID: 哈希算法  
szOID\_RSA\_MD5 = "1.2.840.113549.2.5"  
szOID\_OIWSEC\_sha1 = "1.3.14.3.2.26"

**返回值:** 成功时返回 0,

失败时返回<0, 由 ErrorCode 属性中取错误码, 由 ErrorMessage 属性中取错误信息。

## GetTimeStampVersion

**函数声明:** BSTR GetTimeStampVersion();

**说明:** 获得时间戳版本。前提条件: 已经成功地发送时间戳请求, 并成功地获得时间戳响应。

**参数:**

**返回值:** 成功时返回时间戳版本,

失败时返回空, 由 ErrorCode 属性中取错误码, 由 ErrorMessage 属性中取错误信息。

## GetTimeStampTimeStr

**函数声明:** BSTR GetTimeStampTimeStr ();

**说明:** 获得时间戳签名时间字符串。前提条件: 已经成功地发送时间戳请求, 并成功地获得时间戳响应。

**参数:**

**返回值:** 成功时返回时间戳签名时间字符串,

失败时返回空, 由 ErrorCode 属性中取错误码, 由 ErrorMessage 属性中取错误信息。

## GetTimeStampSerialNumber

**函数声明:** BSTR GetTimeStampSerialNumber ();

**说明:** 获得时间戳签名证书序列号。前提条件: 已经成功地发送时间戳请求, 并成功地获得

时间戳响应。

参数:

返回值: 成功时返回时间戳签名证书序列号,

失败时返回空, 由 ErrorCode 属性中取错误码, 由 ErrorMessage 属性中取错误信息。

## GetTimeStampMessageImprint

函数声明: BSTR GetTimeStampMessageImprint ();

说明: 获得时间戳签名消息拇指印。前提条件: 已经成功地发送时间戳请求, 并成功地获得时间戳响应。

参数:

返回值: 成功时返回时间戳签名消息拇指印,

失败时返回空, 由 ErrorCode 属性中取错误码, 由 ErrorMessage 属性中取错误信息。

## GetTimeStampToken

函数声明: BSTR GetTimeStampToken ();

说明: 获得时间戳响应。前提条件: 已经成功地发送时间戳请求, 并成功地获得时间戳响应。

参数:

返回值: 成功时返回时间戳响应,

失败时返回空, 由 ErrorCode 属性中取错误码, 由 ErrorMessage 属性中取错误信息。

## GetTimeStampSrcHash

函数声明: BSTR GetTimeStampSrcHash ();

说明: 获得时间戳中包含的原文数据哈希值。前提条件: 已经成功地发送时间戳请求, 并成功地获得时间戳响应。

参数:

返回值: 成功时返回时间戳中包含的原文数据哈希值,

失败时返回空, 由 ErrorCode 属性中取错误码, 由 ErrorMessage 属性中取错误信息。

## GetTimeStampSrcAlgo

函数声明: BSTR GetTimeStampSrcAlgo ();

说明: 获得时间戳中包含的原文数据算法。前提条件: 已经成功地发送时间戳请求, 并成功地获得时间戳响应。

参数:

返回值: 成功时返回时间戳中包含的原文数据算法,

失败时返回空, 由 ErrorCode 属性中取错误码, 由 ErrorMessage 属性中取错误信息。

## ViewTimeStampCertWnd

**函数声明:** long ViewTimeStampCertWnd ();

**说明:** 显示时间戳签名证书窗体。前提条件: 已经成功地发送时间戳请求, 并成功地获得时间戳响应。

**参数:**

**返回值:** 成功时返回 0,

失败时返回<0, 由 ErrorCode 属性中取错误码, 由 ErrorMessage 属性中取错误信息。

## 7.14 其他公共方法

### GenUUID

**函数声明:** BSTR GenUUID (long iRetType);

**说明:** 产生 UUID。

**参数:**

- **iRetType:** 输入返回数据的类型
  - 0: 返回两位十六进制数据(中间含 "-" 符号)(36 位长度)
  - 1: 返回两位十六进制数据(中间不含 "-" 符号)(32 位长度)

**返回值:** 成功时返回 BASE64 编码 UUID,

失败时返回空, 由 ErrorCode 属性中取错误码, 由 ErrorMessage 属性中取错误信息。

### CryptGenRandom

**函数声明:** BSTR CryptGenRandom (long iRandLen);

**说明:** 产生随机数。

**参数:**

- **iRandLen:** 指定要产生的随机数长度

**返回值:** 成功时返回 BASE64 编码随机数,

失败时返回空, 由 ErrorCode 属性中取错误码, 由 ErrorMessage 属性中取错误信息。

### BinToBase64

**函数声明:** BSTR BinToBase64 (BSTR sSource, long SourceLen, long IsCert);

**说明:** 将二进制数据转成 BASE64 编码数据, 其中二进制数据长度在网页、VB 等程序调用中请输入二进制数据的字节码长度, 尤其是中文字符, 一个中文字符等于 2 个字节, 可以通过本控件提供的 long StringToByteLength (BSTR sSource) 函数获得长度。

**参数:**

- **sSource:** 二进制数据



- SourceLen: 二进制数据长度(字节)
- IsCert: 指定是否操作证书(0=不是证书,1=是证书)

返回值: 成功时返回 BASE64 编码数据,

失败时返回空, 由 ErrorCode 属性中取错误码, 由 ErrorMessage 属性中取错误信息。

## Base64ToBin

函数声明: BSTR Base64ToBin (BSTR sEncoded, long EncodedLen, long IsCert);

说明: 将 BASE64 编码数据转成二进制编码数据, 其中二进制数据长度在网页、VB 等程序调用中请输入二进制数据的字节码长度, 尤其是中文字符, 一个中文字符等于 2 个字节, 可以通过本控件提供的 long StringToByteLength (BSTR sSource)函数获得长度。

参数:

- sEncoded: BASE64 编码数据
- EncodedLen: BASE64 编码数据长度(字节)
- IsCert: 指定是否操作证书(0=不是证书,1=是证书)

返回值: 成功时返回二进制数据,

失败时返回空, 由 ErrorCode 属性中取错误码, 由 ErrorMessage 属性中取错误信息。

## ReadDataLength

函数声明: long ReadDataLength ();

说明: 读取数据长度, 前提条件: 成功地调用了 Base64ToBin()函数

参数:

返回值: 返回数据长度

## StringToByteLength

函数声明: long StringToByteLength(BSTR Str);

说明: 获得字符串的 Ansi 字节长度

参数:

- Str: 原文数据

返回值: 成功返回原文数据长度, 失败返回 0

## StringToUnicodeLength

函数声明: long StringToUnicodeLength(BSTR Str);

说明: 获得字符串的 Unicode 宽字节长度

参数:

- Str: 原文数据

返回值: 成功返回原文数据长度, 失败返回 0

## AboutBox //吉大兼容

函数声明: void AboutBox ();

说明: 显示该控件的版本窗口, 可以用于判断控件是否可以使用

参数:

返回值: 无

## GetVersion

函数声明: BSTR GetVersion ();

说明: 获得该控件的版本, 通过外部脚本, 可以用于判断控件是否为最新版本, 即是否可以使用。

参数:

返回值: 返回获得该控件的版本

## SetShowError

函数声明: void SetShowError(long IValue);

说明: 设置显示错误信息方式, 该接口函数为在调试时使用而设置的。

默认使用该控件时, 可以不调用此函数。

参数:

- IValue: 状态值

0 = 通过 **ErrorCode** 属性返回错误代码或者 **ErrorMessage** 属性返回错误信息

1 = 除了有 0 状态的功能外, 还可通过消息窗口显示错误信息

返回值: 无

## 8 返回码与错误信息

```
#define E_OK 0L //成功
```

```
#define E_MSG_UNKNOWN_INVALIDATE "未知错误"
```

```
#define E_UNKNOWN_INVALIDATE -6001
```

```
#define E_MSG_PARAM_INVALIDATE "参数错误"
```

```
#define E_PARAM_INVALIDATE -6002
```

```
#define E_MSG_OPEN_SESSION_FAILED "进行网络连接时打开会话失败"
```

```
#define E_OPEN_SESSION_FAILED -6003
```

#define E\_MSG\_CONECT\_TO\_SERVER\_FAILED "进行网络连接时连接服务器失败"

#define E\_CONECT\_TO\_SERVER\_FAILED -6004

#define E\_MSG\_OPEN\_REQUEST\_FAILED "进行网络连接时打开请求失败"

#define E\_OPEN\_REQUEST\_FAILED -6005

#define E\_MSG\_ADD\_HEADER\_FAILED "进行网络连接时添加请求头失败"

#define E\_ADD\_HEADER\_FAILED -6006

#define E\_MSG\_SEND\_REQUEST\_FAILED "发送请求失败"

#define E\_SEND\_REQUEST\_FAILED -6007

#define E\_MSG\_END\_REQUEST\_FALED "结束请求失败"

#define E\_END\_REQUEST\_FALED -6008

#define E\_MSG\_OPEN\_STORE\_FAILED "打开系统证书库失败"

#define E\_OPEN\_STORE\_FAILED -6009

#define E\_MSG\_SELECT\_CERT\_FAILED "选择证书失败"

#define E\_SELECT\_CERT\_FAILED -6010

#define E\_MSG\_CERT\_NOT\_INITIALIZED "证书未初始化"

#define E\_CERT\_NOT\_INITIALIZED -6011

#define E\_MSG\_BUFFER\_TOO\_SMALL "缓冲区太小"

#define E\_BUFFER\_TOO\_SMALL -6012

#define E\_MSG\_CERT\_NOT\_IN\_STORE "找不到指定的证书"

#define E\_CERT\_NOT\_IN\_STORE -6013

#define E\_MSG\_ADD\_CERT\_TO\_STORE "添加证书到证书库失败"

#define E\_ADD\_CERT\_TO\_STORE -6014

#define E\_MSG\_CREAT\_CERT\_FAILED "创建证书上下文失败"

#define E\_CREAT\_CERT\_FAILED -6015

#define E\_MSG\_ALLOC\_MEM\_FAILED "内存不足!"

#define E\_ALLOC\_MEM\_FAILED -6016

#define E\_MSG\_REALLOC\_MEM\_FAILED "重新分配内存失败!"

#define E\_REALLOC\_MEM\_FAILED -6017

#define E\_MSG\_STREAM\_BUFFER\_IS\_NULL "流缓存为空!"

```
#define E_STREAM_BUFFER_IS_NULL      -6018

#define E_MSG_INIT_CERT_NOT_BEGIN_TIME  "证书还未生效!"
#define E_INIT_CERT_NOT_BEGIN_TIME    -6019

#define E_MSG_INIT_CERT_OVER_END_TIME  "证书已经过期!"
#define E_INIT_CERT_OVER_END_TIME     -6020

#define E_MSG_CERT_TYPE_ERROR         "证书类型错误!"
#define E_CERT_TYPE_ERROR             -6021

#define E_MSG_CAN_NOT_FIND_LIBRARY    "找不到动态库"
#define E_CAN_NOT_FIND_LIBRARY        -6022

#define E_MSG_DATA_CHANGE_CODE_FAILED "数据转码失败"
#define E_DATA_CHANGE_CODE_FAILED     -6023

#define E_MSG_CRYPT_FAILED            "密码类无效"
#define E_CRYPT_FAILED                -6024

#define E_MSG_SYM_CRYPT_FAILED        "对称密码类无效"
#define E_SYM_CRYPT_FAILED            -6025

#define E_MSG_VERIFY_HASH_FAILED      "验证哈希失败"
#define E_VERIFY_HASH_FAILED          -6026
```

其他返回码和错误信息为操作系统函数返回信息，可以查询微软帮助。