LNCA-CryptoAPI-Com 程序员手册 v1.6.5.0

辽宁省数字证书认证中心
Liaoning Certificate Authority Center

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

目　录

1 引言

1.1 编写目的

本手册由 LNCA-CryptoAPI 项目开发人员撰写，主要目的在于向用户介绍有关安全和 PKI 方面的一些概念、LNCA-CryptoAPI 的设计思路和详细使用方法，使得用户能够理解并正确使用 LNCA-CryptoAPI 所提供的功能，从而正确并高效地将辽宁省数字证书认证中心颁发的数字证书通过 LNCA-CryptoAPI 应用于自己的应用系统中，最终达到使应用系统具有所需系统安全性的目的。

本用户手册的读者对象为辽宁省数字证书认证中心从事技术支持工作的相关软件工程师和使用 PKI 技术实现系统信息安全的单位中从事应用系统开发工作的软件技术人员。

1.2 背景

近年来，Internet 在中国得到了迅速发展，基于 Web 的应用软件系统逐渐成为新一代应用软件系统的主流。人们在开发和使用基于 Web 的应用软件系统的过程中，对系统应具有的信息安全性提出了越来越高的要求，因此，国际上公认的、能够为应用软件系统提供实用信息安全性的 PKI 相关技术得到了越来越多的应用。

辽宁省数字证书认证中心在为广大用户提供信息安全服务的过程中，发现由 CA 系统开发商提供的现有 CryptoAPI 不能完全满足很多应用软件系统对信息安全性的要求，这些问题对信息安全技术在辽宁省范围内迅速得到广泛应用将造成一定影响。为了解决现存 CryptoAPI 存在的这些问题、提高服务质量、促进信息安全技术在辽宁省范围内迅速得到广泛应用，辽宁省数字证书认证中心软件部在对应用软件系统在实现信息安全方面的需求进行认真研究的基础上开发了 LNCA-CryptoAPI。

1.3 设计目标

- 弥补现存 CryptoAPI 的不足，使 LNCA-CryptoAPI 支持 USB-KEY 和加密机（加密卡）两种硬件证书容器，确保应用客户端和服务器端证书都具有较高的安全性；
- 为应用系统开发提供统一的接口，以提高应用系统的可维护性。

1.4 参考资料

资料名称	作者	发布日期
《JIT 应用 ToolkitsAPI（COM 版）程序员手册》	吉大正元	
《JITCertToolks 控件程序员手册》	吉大正元	

表 1 参考资料表

2 PKI 简介

PKI 是指公钥基础设施，它是目前国际上公认的较成熟和较完善的一组信息安全技术标准，主要包括如下内容。

2.1 对称数据加密技术

对称数据加密技术由对称数据加密算法和相应的密钥生成算法组成，对称数据加密算法种类较多，常用的有 DES 系列、RC 系列及 SSF33 算法等。对称数据加密算法在进行数据加、解密操作时通常使用相同的密钥，这就是“对称”的由来。

对称数据加密算法的特点是运算速度较快，适合于大量数据进行操作。但由于数据加、解密操作时使用相同的密钥，在通信双方之间安全地交换密钥是在实际应用中最难解决的问题。

2.2 非对称数据加密技术

非对称数据加密技术由非对称数据加密算法和相应的密钥生成算法组成，非对称数据加密算法种类较少，目前最常用的是RSA算法。非对称数据加密算法采用一对相互匹配的密钥分别用于数据加、解密操作，这对密钥分别称为“公钥”和“私钥”。在使用非对称数据加密算法进行数据加、解密操作时，若使用公钥进行数据加密，必须使用与之匹配的私钥进行数据解密，反之亦然。由于这些算法在进行数据加、解密时使用不同的密钥，故称之为“非对称加密算法”。

非对称加密算法的特点是数据加、解密操作的速度比对称加密算法慢很多，不适合于对大量数据进行操作。但它们却是在安全数据通信过程中交换对称加密密钥的理想技术。

在将非对称数据加密技术应用于安全的数据通信中时，每个参与数据通信的实体拥有自己的非对称加密算法密钥对，私钥由其拥有者妥善保管并应采取有效的措施防止其它任何实体得到它，公钥则可以任意复制给参与数据通信的其它实体。假设有A、B两个实体要进行安全的数据通信，实体A要将自己的数据（称为原文）发送给实体B，则实体A首先使用实体B的公钥对自己的原文进行非对称数据加密处理（产生的结果称为密文），然后实体A可以采用任何数据传输方式将密文发送给实体B，实体B使用自己的私钥对接收到的密文进行非对称数据解密处理即可得到实体A所发送的原文。若在密文传输过程中，实体C有意或无意地获取了密文，只要实体C不能得到实体B的私钥，非对称数据加密算法可以保证在合理的时间内实体C不能破解出原文而泄密。

2.3 数据摘要技术

数据摘要技术仅由数据摘要算法组成，数据摘要算法种类较多，目前常用的有RC4、MD5、SHA－1等。数据摘要算法是一种单向数据特征汇集函数，具有下列特点：

① 对相同的数据进行处理总是产生相同的结果；

② 对不同的数据进行处理总是产生不同的结果；

③ 处理结果对原始数据的差异非常敏感；
④ 同一算法对任意长度数据处理产生的结果长度固定且通常很小。
⑤ 从处理结果不能计算出原始数据。

由于数据摘要算法具有上述特征，人们常称数据摘要算法的处理结果为原始数据的“指纹”。

2.4 数字签名技术

数字签名技术是数据摘要技术和非对称数据加密技术的综合应用。对数据进行数字签名操作需要依次进行下列两步处理：

① 对数据进行数据摘要处理；
② 对上步的处理结果使用私钥进行非对称数据加密操作。

数字签名操作的最终结果称为原始数据的“数字签名”。

与数字签名操作相反的操作是验证签名操作。验证签名操作需要依次进行下列三步处理：

① 使用与私钥相匹配的公钥对数字签名进行非对称数据解密操作；
② 对原始数据进行数据摘要处理；
③ 将上两步的结果进行比较，比较结果相同表示数字签名验证成功。

2.5 公钥证书标准

公钥证书标准用于规范数字证书的组成和数据格式，使得符合标准的数字证书可以应用于不同的PKI技术实现产品中。数字证书用于记载证书拥有者（称为证书主题）的公钥及与证书主题和证书颁发者相关的一些信息，数字证书的内容由证书颁发者使用其私钥进行数字签名以保证所记载数据的真实性、完整性。

2.6 数字信封标准

数字信封标准用于规定方便使用的通用数据交换格式，常用数字信封有以下几种：

2.6.1 签名信封

签名信封应用数字签名技术用于发送者向接收者表明其所发送信息的真实性和完整性。签名信封由数据原文、数字签名、签名者数字证书和所采用的数字签名算法等信息组成，签名信封的接收者可以验证签名信封的有效性并得到数据原文。

2.6.2 加密信封

加密信封综合运用对称数据加密技术和非对称数据加密技术，用于发送者以保证信息保密性的方式向接收者传递敏感信息。加密信封由数据原文经对称数据加密处理产生的数据密文、对称数据加密密钥经非对称数据加密处理产生的密钥密文和所采用的对称

加密算法等信息组成，接收者可以使用自己的私钥解密加密信封并得到数据原文。

2.6.3 签名加密信封

签名加密信封综合运用数字签名技术、对称数据加密技术和非对称数据加密技术，用于发送者以保证信息保密性、真实性和完整性的方式向接收者传递敏感信息。签名加密信封由数据原文经对称数据加密处理产生的数据密文、数字签名经对称数据加密处理产生的签名密文、对称数据加密密钥经非对称数据加密处理产生的密钥密文、签名者数字证书、所采用的数字签名算法和对称加密算法等信息组成，签名加密信封的接收者可以使用自己的私钥解密验证签名加密信封的有效性并得到数据原文。

2.7 私钥管理技术

私钥管理技术是采用PKI安全体系实现系统信息安全的保障，如果私钥不能安全地保管和使用，上述任何技术都不能提供信息安全保证。目前，广泛采用的私钥管理技术有USB－KEY、智能卡、加密卡和加密机等硬件设备。这些设备都具有产生密钥对或通过安全的方式导入密钥对、导出公钥、通过自带的CPU使用私钥进行非对称加、解密等功能。请注意：私钥只能通过安全的方式导入而不能导出。

3 LNCA-CryptoAPI(Com 版)的组成

LNCA-Crypto(Com版)目前支持两种接口引擎(SoftDevice_COM, HardDevice_COM)。以下各节将对每种LNCA-CryptoAPI接口引擎的作用和使用时所需的配置信息进行详细介绍。

3.1 LNCA-CryptoAPI(Com 版)接口 SoftDevice_COM 引擎

3.1.1 功能

3.1.1.1 支持的证书容器类型

SoftDevice_COM引擎支持PKCS12文件证书容器。每个PKCS12文件中可以存放一个证书对象。

3.1.1.2 支持的摘要算法

SoftDevice_COM引擎支持MD2、MD4、MD5和SHA-1四种摘要算法。对应的算法OID值为:
MD2:　"1.2.840.113549.2.2"　= szOID_RSA_MD2

MD4: "1.2.840.113549.2.4" = szOID_RSA_MD4
MD5: "1.2.840.113549.2.5" = szOID_RSA_MD5
SHA-1: "1.3.14.3.2.26" = szOID_OIWSEC_sha1

3.1.1.3 支持的签名算法

SoftDevice_COM引擎支持MD2WithRSA、MD5WithRSA和SHA1WithRSA三种签名算法。对应的算法OID值为:

MD2: "1.2.840.113549.2.2" = szOID_RSA_MD2
MD5: "1.2.840.113549.2.5" = szOID_RSA_MD5
SHA-1: "1.3.14.3.2.26" = szOID_OIWSEC_sha1

3.1.1.4 支持的MAC算法

SoftDevice_COM引擎目前不支持MAC操作。

3.1.1.5 支持的对称加密算法

SoftDevice_COM引擎支持3DES、DES、RC4、RC2四种对称加密算法。对应的算法OID值为:

3DES(CALG_3DES) : "1.2.840.113549.3.7"
DES(CALG_DES) : "1.3.14.3.2.7"
RC4(CALG_RC4): "1.2.840.113549.3.4"
RC2(CALG_RC2): "1.2.840.113549.3.2"

3.1.1.6 支持的非对称加密算法

SoftDevice_COM引擎只支持RSA非对称加密算法。对应的算法OID值为:

3DES: "1.2.840.113549.3.7" = szOID_RSA_DES_EDE3_CBC
DES_CBC: "1.3.14.3.2.7" = szOID_OIWSEC_desCBC
RC4: "1.2.840.113549.3.4" = szOID_RSA_RC4
RC2_CBC: "1.2.840.113549.3.2" = szOID_RSA_RC2CBC

3.1.1.7 支持的数字信封类型

SoftDevice_COM引擎支持签名信封、加密信封和签名加密信封三种数字信封。对于签名加密信封支持PKCS7标准一种实现。

3.1.2 文件列表

- LNCACryptoCom.dll
- LNCACryptoCom.tlb
- pkilib.dll
- ProofTime.dll
- cacipher.ini
- lnca_crypto.cfg

3.2 LNCA-CryptoAPI(Com 版)接口 HardDevice_COM 引擎

3.2.1 功能

3.2.1.1 支持的证书容器类型

HardDevice_COM引擎支持加密机（加密卡）证书容器。每个加密机（加密卡）中可以存放多个证书对象。

3.2.1.2 支持的摘要算法

HardDevice_COM引擎支持MD5和SHA-1两种摘要算法。对应的算法OID值为:

MD5: "1.2.840.113549.2.5" = szOID_RSA_MD5

SHA-1: "1.3.14.3.2.26" = szOID_OIWSEC_sha1

3.2.1.3 支持的签名算法

HardDevice_COM引擎支持MD5WithRSA和SHA1WithRSA两种签名算法。对应的算法OID值为:

MD5: "1.2.840.113549.2.5" = szOID_RSA_MD5

SHA-1: "1.3.14.3.2.26" = szOID_OIWSEC_sha1

3.2.1.4 支持的MAC算法

56Cipher_COM引擎目前不支持MAC操作。

3.2.1.5 支持的对称加密算法

HardDevice_COM引擎目前不支持加密机的对称加密操作,而只支持微软的3DES、DES、RC4、RC2四种对称加密算法。对应的算法OID值为:

3DES(CALG_3DES) ： "1.2.840.113549.3.7"

DES(CALG_DES) ： ″1.3.14.3.2.7″
RC4(CALG_RC4)： ″1.2.840.113549.3.4″
RC2(CALG_RC2)： ″1.2.840.113549.3.2″

3.2.1.6 支持的非对称加密算法

HardDevice_COM引擎只支持RSA非对称加密算法。对应的算法OID值为:

3DES:	"1.2.840.113549.3.7"	= szOID_RSA_DES_EDE3_CBC
DES_CBC:	"1.3.14.3.2.7"	= szOID_OIWSEC_desCBC
RC4:	"1.2.840.113549.3.4"	= szOID_RSA_RC4
RC2_CBC:	"1.2.840.113549.3.2"	= szOID_RSA_RC2CBC

3.2.1.7 支持的数字信封类型

HardDevice_COM引擎支持签名信封、加密信封和签名加密信封三种数字信封。对于签名加密信封支持PKCS7标准一种实现。

3.2.2 文件列表

- LNCACryptoCom.dll
- LNCACryptoCom.tlb
- pkilib.dll
- ProofTime.dll
- cacipher.ini
- lnca_crypto.cfg

4 LNCA-CryptoAPI(Com 版)使用指南

4.1 注意事项

- COM 组件内部返回的错误码均为负数。
- 引擎接口对象中的 CreateEngine()函数是每个引擎接口对象所必须调用的。在初始化函数时，要指定工作路径，在工作路径下应当有一个 lnca_crypto.cfg 配置文件，此配置文件是 COM 组件运行时所必须有的，文件内部的信息是 COM 组件运行时需要用到的，因此必须正确设置。
- 证书接口对象中的 SetEngineInfo()函数是每个证书接口对象所必须调用的。目的是从引擎接口对象中获得配置文件路径，初始化根证书和 CRL。
- COM 组件内部维护着一个证书动态数组，用户在使用引擎接口对象时必须先调用 CreateEngine()函数，成功地初始化引擎对象之后，数组中的相应元素保存证书对象。其中的编号顺序为：

0 = 根证书，默认为辽宁 CA
1 = 返回签名信封中的签名者证书，初始时为空
2 = 返回加密信封中的接收人证书，初始时为空
3 = 返回时间戳包含的证书，初始时为空
4 = 根据配置文件中的设置，初始化第一个服务器证书
…以后的编号根据配置文件配置的不同，可能有多个服务器证书或者用户公钥证书等。

其中证书数组编号 1,2,3 是预先保留的内部证书对象，当调用了相应的操作函数之后，上述 3 项才有值。

要想使用其它的证书，是通过证书初始化函数 CreateCertificateFromFile()、CreateCertificateFromMem()和 CreateCertificateFromObject ()来完成的。

4.2 COM 组件配置

4.2.1 加密机配置

当 COM 组件使用的服务器证书为加密机，应当使用加密机 cacipher.ini 的配置文件。
cacipher.ini：密码机配置文件，内设允许访问密码机的服务器（主机）的 IP 地址；
cacipher.ini 文件应当放在运行目录或操作系统目录下。
如：c:\WINNT\system32\drivers\etc\cacipher.ini

4.2.2 COM 接口引擎配置

在 LNCA-CryptoAPI 接口引擎中，通过一个文件名为 lnca_crypto.cfg 的配置文件来指定引擎所需的一些配置参数。每一个配置参数由配置文件的一行来指定，其格式为：
<key>=<value>
<key>表示配置参数的标识名，与 Window INI 格式规则相同；
<value>表示配置参数的实际值。

标识符	参数值
[cryptography]	
errorReturnType=0	运行时错误返回方式: 0 = 通过函数返回错误代码或者错误信息 1 = 通过消息窗口显示错误信息 2 = 通过日志文件保存错误信息 注：在 B/S 环境下，此参数必须为 0.
isValidCertTime=1	是否效验证书有效期: 0 = 不效验证书有效期 1 = 效验证书有效期
rootCertificateFile=c:/cert/LNCA_bin.cer	指定公钥根证书文件(全路径)(二进制编码)
crlFile= c:/cert /LNCA.crl	指定 CRL 文件(全路径)(二进制编码)
crlVerifyMode=1	指定 CRL 效验方式 0 = 在线，1 = 离线

[serverType]	
serverTypeCount=2	服务器类型总数
currentServerType=1	当前使用的服务器类型编号 1 = 微软.PFX 服务器引擎类型 2 = 加密机或加密卡服务器引擎类型
serverType1=SoftDevice_COM name1=PKCS12_SYSTEM serverType2=HardDevice_COM name2=加密机或加密卡	此为服务器类型信息，通过此信息可以对应找到下面具体的段信息，可以根据具体服务器类型进行配置
[SoftDevice_COM]	
serverSoftCertType = Store	服务器软证书类型： Store = 从服务器系统证书库获得证书，File = 从下面参数指定.pfx 文件证书
keySetType = 1	服务器软证书保存在在系统的位置： 1 = CERT_SYSTEM_STORE_LOCAL_MACHINE 2 = CERT_SYSTEM_STORE_CURRENT_USER 3 = CERT_SYSTEM_STORE_USERS 4 = CERT_SYSTEM_STORE_SERVICES 5=CERT_SYSTEM_STORE_CURRENT_SERVICE 注：当 serverSoftCertType = File 时，此项只能为 keySetType = 1
systemStorePos = MY	服务器 System store 位置 可选参数： systemStorePos = MY systemStorePos = IISADMIN\MY systemStorePos = W3SVC\MY 注：此项当 serverSoftCertType = Store 时有效
certCount = 2	服务器软证书数量
storeCertSN1= f1e011e6f90b3ce4c931280062c2009 …	从服务器系统证书库枚举要使用的证书序列号 此参数可以指定多个证书序列号，目的是实现 USBKey 双证书的初始化功能。
serverCertificateFile1=c:/cert/www.demo.com.pfx …	服务器软证书(.pfx) (全路径) 此参数可以指定多个证书文件，目的是实现多服务器证书的初始化功能。
serverCertificatePassword=11111111	服务器证书保护口令
serverCertificateIsBase64Code=0	服务器证书编码类型 0 = 二进制编码 1 = BASE64 编码
[HardDevice_COM]	

slotId=1	在加密机（加密卡）中所使用设备的标识号
userPIN=88880001	在加密机（加密卡）中所使用设备的普通用户通行字
keyID=28	在加密机（加密卡）中所使用设备 key ID 号
[cert_extension]	
certificate.extensionCount=6	证书扩展域总数
extension.oid1=1.2.86.11.7.1 extension.id1=identifyCard extension.name1=个人身份标识码 extension.oid2=1.2.86.11.7.2 extension.id2=insuranceNumber extension.name2=个人社会保险号 extension.oid3=1.2.86.11.7.3 extension.id3=organizationCode extension.name3=企业组织机构代码 extension.oid4=1.2.86.11.7.4 extension.id4=registerNumber extension.name4=企业工商注册号 extension.oid5=1.2.86.11.7.5 extension.id5=countryTaxNumber extension.name5=企业(国税)号 extension.oid6=1.2.86.77.1 extension.id6=localTaxMnuber extension.name6=企业(地税)号	此为证书扩展域信息，可以根据具体证书扩展域项进行配置
[SoftDevice_COM _algorithm]	
digestCount=4	摘要算法总数
digest.id1=SHA1 digest.number1=544 digest.oid1=1.3.14.3.2.26 digest.name1=szOID_OIWSEC_sha1 digest.id2=MD5 digest.number2=528 digest.oid2=1.2.840.113549.2.5 digest.name2=szOID_RSA_MD5 digest.id3=MD4	此为摘要算法信息，可以根据具体摘要算法进行配置

digest.number3= digest.oid3=1.2.840.113549.2.4 digest.name3=szOID_RSA_MD4 digest.id4=MD2 digest.number4=512 digest.oid4=1.2.840.113549.2.2 digest.name4=szOID_RSA_MD2	
signCount=3	签名算法总数
sign.id1=SHA1 sign.number1=6 sign.oid1=1.3.14.3.2.26 sign.name1=szOID_OIWSEC_sha1 sign.id2=MD5 sign.number2=5 sign.oid2=1.2.840.113549.2.5 sign.name2=szOID_RSA_MD5 sign.id3=MD2 sign.number3=4 sign.oid3=1.2.840.113549.2.2 sign.name3=szOID_RSA_MD2	此为签名算法信息，可以根据具体签名算法进行配置
rsaEncryptCount=4	非对称加密算法总数
rsaEncrypt.id1=DES3/CBC rsaEncrypt.number1=307 rsaEncrypt.oid1=1.2.840.113549.3.7 rsaEncrypt.name1=szOID_RSA_DES_EDE3_CBC rsaEncrypt.id2=DES/CBC rsaEncrypt.number2=290 rsaEncrypt.oid2=1.3.14.3.2.7 rsaEncrypt.name2=szOID_OIWSEC_desCBC rsaEncrypt.id3=RC4 rsaEncrypt.number3= rsaEncrypt.oid3=1.2.840.113549.3.4 rsaEncrypt.name3=szOID_RSA_RC4 rsaEncrypt.id4=RC2/CBC rsaEncrypt.number4=258	此为非对称加密算法信息，可以根据具体非对称加密算法进行配置

rsaEncrypt.oid4=1.2.840.113549.3.2 rsaEncrypt.name4=szOID_RSA_RC2CBC	
macCount=0	MAC 算法总数
symCipherCount=4	对称加密算法总数
symCipher.id1=3DES symCipher.number1=CALG_3DES symCipher.oid1=1.2.840.113549.3.7 symCipher.name1= symCipher.id2=DES symCipher.number2=CALG_DES symCipher.oid2=1.3.14.3.2.7 symCipher.name2= symCipher.id3=RC4 symCipher.number3=CALG_RC4 symCipher.oid3=1.2.840.113549.3.4 symCipher.name3= symCipher.id4=RC2 symCipher.number4=CALG_RC2 symCipher.oid4=1.2.840.113549.3.2 symCipher.name4=	此为对称加密算法信息，可以根据具体对称加密算法进行配置
[HardDevice_COM _algorithm]	
digestCount=2	摘要算法总数
digest.id1=SHA1 digest.number1=544 digest.oid1=1.3.14.3.2.26 digest.name1=szOID_OIWSEC_sha1 digest.id2=MD5 digest.number2=528 digest.oid2=1.2.840.113549.2.5 digest.name2=szOID_RSA_MD5	此为摘要算法信息，可以根据具体摘要算法进行配置
signCount=2	签名算法总数
sign.id1=SHA1 sign.number1=6 sign.oid1=1.3.14.3.2.26 sign.name1=szOID_OIWSEC_sha1	此为签名算法信息，可以根据具体签名算法进行配置

sign.id2=MD5 sign.number2=5 sign.oid2=1.2.840.113549.2.5 sign.name2=szOID_RSA_MD5	
rsaEncryptCount=4	非对称加密算法总数
rsaEncrypt.id1=DES3/CBC rsaEncrypt.number1=307 rsaEncrypt.oid1=1.2.840.113549.3.7 rsaEncrypt.name1=szOID_RSA_DES_EDE3_CBC rsaEncrypt.id2=DES/CBC rsaEncrypt.number2=290 rsaEncrypt.oid2=1.3.14.3.2.7 rsaEncrypt.name2=szOID_OIWSEC_desCBC rsaEncrypt.id3=RC4 rsaEncrypt.number3= rsaEncrypt.oid3=1.2.840.113549.3.4 rsaEncrypt.name3=szOID_RSA_RC4 rsaEncrypt.id4=RC2/CBC rsaEncrypt.number4=258 rsaEncrypt.oid4=1.2.840.113549.3.2 rsaEncrypt.name4=szOID_RSA_RC2CBC	此为非对称加密算法信息，可以根据具体非对称加密算法进行配置
macCount=0	MAC 算法总数
symCipherCount=4	对称加密算法总数
symCipher.id1=3DES symCipher.number1=CALG_3DES symCipher.oid1=1.2.840.113549.3.7 symCipher.name1= symCipher.id2=DES symCipher.number2=CALG_DES symCipher.oid2=1.3.14.3.2.7 symCipher.name2= symCipher.id3=RC4 symCipher.number3=CALG_RC4 symCipher.oid3=1.2.840.113549.3.4 symCipher.name3=	此为对称加密算法信息，可以根据具体对称加密算法进行配置

symCipher.id4=RC2 symCipher.number4=CALG_RC2 symCipher.oid4=1.2.840.113549.3.2 symCipher.name4=	

4.3 编程指南综述

LNCA-CryptAPI 的设计目标之一是为使用 PKI 技术实现信息安全的应用系统开发提供统一的高级 CryptoAPI 接口，以提高应用系统的可维护性。为了实现这一目标，我们在 LNCA-CryptAPI Com 库中定义了 LNCAEngineCom、LNCACertificateCom 接口。

LNCAEngineCom 接口：
ProgID = LNCACryptoCom.LNCAEngineCom.1
UUID = 2B5F9ED5BA6E4B8183025FFEB10816DC
LNCACertificateCom 接口：
ProgID = LNCACryptoCom.LNCACertificateCom.1
UUID = C0A367A7328345CA8C223EF2B945DA6D

LNCA-CryptoAPI 目前有 SoftDevice_COM 和 HardDevice_COM 两种引擎，以下几节将详细介绍这几个接口的作用和使用方法。

4.4 LNCAEngineCom 接口

位于 Com 库的 LNCAEngineCom 接口用于定义密码系统引擎实现类应提供的功能。每个密码系统引擎对象实际上是一个软件加密机运行环境，它完成下列功能：

- 构造密码系统中的各种证书对象；
- 为其构造的各种证书对象提供运行环境；
- 实现几种公共操作。

在使用 LNCA-CryptoAPI 时应注意下面几个原则：

- ✧ 在一个应用程序中，每种密码系统引擎最多只能有一个实例
- ✧ 应用程序在构造 LNCAEngineCom 对象后，必须通过执行其 CreateEngine()方法进行正确初始化

4.4.1 引擎相关属性

InputDataType　输入数据格式类型

属性声明：

获得：long　InputDataType()

设置：InputDataType(long newVal)

属性说明：

获得或者设置输入的数据格式类型，在其他函数中用来配合控制输入数据或者文件数据的格式。

0 = 输入二进制格式，不进行转码；

1 = 输入 BASE64 编码，进行转码。

参数说明：

newVal:　　输入数据或者文件数据的格式

返回参数：

OutputDataType　输出数据格式类型

属性声明：

获得：long　OutputDataType()

设置：OutputDataType(long newVal)

属性说明：

获得或者设置输出的数据格式类型，在其他函数中用来配合控制输出数据或者文件数据的格式。

0 = 输出二进制格式，不进行转码；

1 = 输出 BASE64 编码，进行转码。

参数说明：

newVal:　　输出数据或者文件数据的格式

返回参数：

ShowErrorMode　显示错误模式

属性声明：

获得：long　ShowErrorMode ()

设置：ShowErrorMode (long newVal)

属性说明：

获得或设置显示错误模式，该接口属性为在调试时使用而设置的。

默认时，可以不调用此属性。

注：此属性在 B/S 结构中作为 Web 服务器脚本中不能使用。

参数说明：

lValue：状态值

0 = 通过 ErrorCode()方法返回错误代码或者 GetErrorMessage()方法返回错误信息

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：

4.4.2 引擎初始化相关方法

CreateEngine 引擎初始化

方法声明：

long CreateEngine(BSTR sWorkPath, long ServerType);

方法说明：

CreateEngine 方法用于初始化引擎对象。密码系统引擎对象只有在正确初始化之后才能进行其他操作。

对一个成功初始化过的密码系统引擎之后，再次执行初始化操作没有任何作用。

参数说明：

sWorkPath: 配置文件完整路径。

ServerType: 引擎类型 [可选参数]，默认：1 = SoftDevice_COM

在配置文件中:

[serverType]

currentServerType=1 服务器引擎类型序号。在整个应用系统中，引擎类型序号可以通过配置文件找到对应的引擎名称，用于在系统配置中确定初始化引擎对象所需的一组参数。

1 = SoftDevice_COM (即使用微软密码 API)

2 = HardDevice_COM (即使用加密机 API)

返回参数：

成功返回>0，即引擎数组中已经初始化的证书总数，失败返回<=0 值。

CreateCertificate 根据证书路径初始化证书

方法声明：

```
long CreateCertificate(BSTR sDevicePath,
                       BSTR sPassword,
                       long nCertIndex ,
                       long IsBase64Format,
                       BSTR sScription)
```

方法说明：

该方法根据指定的证书路径及通行字构造并初始化一个证书对象。前提条件：已经成功初始化引擎。

证书路径是用于指定证书存储位置的描述串，它由存储介质描述符和文件路径两个部分组成，存储介质描述符和文件路径之间以双斜杠（//）分隔。存储介质描述符有下列两种：

✧ token:

表示证书的存储介质是加密机或加密卡。加密机或加密卡是服务器或设备证书的常用存储介质。

✧ file:

表示证书的存储介质是磁盘文件。磁盘文件是公钥证书和 PKCS12 证书采用的存储介质。

对于不同的存储介质，文件路径的含义也不同。下面是与两种存储介质描述符相对应的文件路径：

✧ 证书标识号

这种文件路径形式与加密机或加密卡存储介质描述符相对应。在一个加密机或加密卡中可以存放多个单证书对象，每个证书对象与一个证书标识号相关联。在“证书标识号”这种文件路径形式中，表示所构造的证书对象是存储在加密机或加密卡中由证书标识号规定的证书对象的代理对象。

✧ 证书文件的绝对路径

这种文件路径形式与磁盘文件存储介质描述符相对应。证书文件的常用扩展名有.cer 和.pfx 两种，前者为公钥证书文件，后者为包含私钥的 PKCS12 格式证书文件。

参数说明：

sDevicePath: 证书信息(字符串不区分大小写)

例如：

TOKEN://1 存储在加密机的证书(1 为 KeyID 号,依据具体情况变化)

FILE://xxx.cer 公钥文件证书

FILE://xxx.pfx PKCS12 文件证书

sPassword: 访问 PKCS12 格式文件证书或加密机或加密卡证书所需的通行字，构造其它形式的证书对象时，应为””。

long nCertIndex 指定初始化证书后保存的位置

<0 或者 >=证书数组总数 ：在证书数组最后追加证书对象,

0 至 证书数组总数-1 范围时 ：替换证书数组中的某个证书对象.

IsBase64Format: 是否为 BASE64 编码格式 [可选参数]，默认：0

0 = 二进制编码，1 = BASE64 编码

sScription: 描述信息 [可选参数]，默认：””，用于帮助记忆的一个字符串

返回参数：

成功返回>=0，失败返回<0 值。返回当前初始化证书在此引擎系统中的编号。

CreateCertificateFromMem 根据证书编码数据初始化证书

方法声明：

```
long CreateCertificateFromMem (BSTR sCertData,
                               BSTR sPassword,
                               long nCertIndex ,
                               long IsBase64Format,
```

long lCertType,
BSTR sScription)

方法说明：

根据指定的编码格式的证书数据构造并初始化公钥证书对象。前提条件：已经成功初始化引擎。

参数说明：

sCertData: 公钥证书编码数据

sPassword: 访问 PKCS12 格式文件证书所需的通行字，构造其它形式的证书对象时，应为""。

long nCertIndex 指定初始化证书后保存的位置

<0 或者 >=证书数组总数 ：在证书数组最后追加证书对象，

0 至 证书数组总数-1：替换证书数组中的某个证书对象.

IsBase64Format: 是否为 Base64 编码格式 [可选参数]，默认：0

0 = 二进制编码，1 = BASE64 编码

lCertType: 证书类型 [可选参数]，默认：0

0 = .CER(公钥证书文件数据)

1 = .PFX(PKCS#12 证书文件数据)

sScription: 描述信息 [可选参数]，默认：""，用于帮助记忆的一个字符串

返回参数：

成功返回>=0，失败返回<0 值。返回当前证书在此引擎系统中的编号。

CreateCertificateFromStore 从服务器系统证书库初始化证书

方法声明：

long CreateCertificateFromStore (BSTR sCertSN,
long nCertIndex,
BSTR sStoreName,
long KeysetType,
BSTR sScription)

方法说明：

根据指定的服务器证书序列号构造并初始化证书对象。前提条件：已经成功初始化引擎。

参数说明：

sCertSN: 要初始化证书的序列号

long nCertIndex 指定初始化证书后保存的位置

<0 或者 >=证书数组总数 ：在证书数组最后追加证书对象，

0 至 证书数组总数-1：替换证书数组中的某个证书对象.

sStoreName: 指定系统证书库名称

My，AddressBook，CA，Root

KeysetType: 密钥存储类型

1= 为 CERT_SYSTEM_STORE_LOCAL_MACHINE

2 = 为 CERT_SYSTEM_STORE_CURRENT_USER

sScription: 描述信息 [可选参数]，默认：""，用于帮助记忆的一个字符串

返回参数：

成功返回>=0，失败返回<0 值。返回当前证书在此引擎系统中的编号。

CreateCertificateFromObject 根据证书接口对象初始化证书

方法声明：

long CreateCertificateFromObject (LPDISPATCH I_Cert,
long nCertIndex ,
BSTR s sScription)

方法说明：

根据指定的证书接口对象构造并初始化公钥证书对象。前提条件：已经成功初始化引擎。

参数说明：

I_Cert: 证书接口对象

long nCertIndex 指定初始化证书后保存的位置

<0 或者 >=证书数组总数 ：在证书数组最后追加证书对象,

0 至 证书数组总数-1：替换证书数组中的某个证书对象.

sScription: 描述信息 [可选参数]，默认：””，用于帮助记忆的一个字符串

返回参数：

成功返回>=0，失败返回<0 值。通过 ReturnLong()函数返回当前证书在此引擎系统中的编号。

EngineCertCount 获得引擎中的证书总数

方法声明：

long EngineCertCount()

方法说明：

获得引擎中的证书总数，前提条件：已经成功初始化引擎。

参数说明：

返回参数：

成功返回证书总数。

ExportPublicCertData 获得公钥证书数据

方法声明：

BSTR ExportPublicCertData(long nCertIndex, long IsBase64Format)

方法说明：

获得引擎中的公钥证书数据，前提条件：已经成功初始化引擎。

参数说明：

nCertIndex: 引擎中的证书编号

IsBase64Format: 是否返回 BASE64 编码格式数据

0=二进制编码，1= BASE64 编码

返回参数：

成功返回编码数据，失败返回 NULL。通过 ReturnLong()函数返回公钥证书数据长度。

CertInitStatus 获得证书初始化状态

方法声明：

long CertInitStatus (long Index);

方法说明：

判断组件内指定证书是否已初始化。

参数说明：

Index：证书索引号

返回参数：

成功返回 0，其他为未初始化。

CertScription 获得证书描述信息

方法声明：

BSTR CertScription (long Index);

方法说明：

获得组件内初始化证书时指定证书的描述信息，用于帮助记忆。如果在初始化时，不指定描述信息，则为空。

参数说明：

Index：证书索引号

返回参数：

返回描述信息，空可能无描述信息或者证书为未初始化，可以通过由 GetErrorCode 方法中取错误码，由 GetErrorMessage 方法中取错误信息。

RemoveCert 清除组件内证书

方法声明：

long RemoveCert (long Index);

方法说明：

清除组件内证书数组中指定的证书，证书数组中根证书:下标[0] 、证书下标[1]、证书下标[2]、证书下标[3]除外。前提条件：已经成功初始化证书。

参数说明：

Index：证书索引号，如果 Index = -1，清除组件内所有证书，其中：证书下标[0]、证书下标[1]、证书下标[2]、证书下标[3]除外。

返回参数：

成功返回 0，

失败时返回<0，由 GetErrorCode 方法中取错误码，由 GetErrorMessage 方法中取错误信息。

GetCertificateProviderName　　获得证书提供者

方法声明：

BSTR GetCertificateProviderName (long nCertIndex);

方法说明：

获得证书的提供者。

注：此函数在 B/S 结构中作为 Web 服务器脚本中只能获得.PFX 文件证书，其它如：加密机、加密卡、.CER 文件证书等无法获得证书的提供者。

参数说明：

nCertIndex:　　指定证书索引号

返回参数：

成功返回提供者，失败返回 NULL。

CompareCertificateProviderName　　比较证书提供者

方法声明：

long CompareCertificateProviderName (long nCertIndex, BSTR CompareStr);

方法说明：

通过指定的提供者字符串比较证书的提供者。

注：此函数在 B/S 结构中作为 Web 服务器脚本中只能获得.PFX 文件证书，其它如：加密机、加密卡、.CER 文件证书等无法获得证书的提供者。

参数说明：

nCertIndex:　　指定证书索引号

CompareStr:　　指定的提供者

返回参数：

成功返回 0，失败返回<0 值。

ViewCertWnd 显示指定证书窗体

方法声明：

long ViewCertWnd (long nCertIndex)

方法说明：

显示证书窗体，前提条件：已经成功初始化引擎，初始化证书。

注：此函数在 B/S 结构中作为 Web 服务器脚本中不能使用。

参数说明：

nCertIndex:　　引擎中的证书编号

返回参数：

成功返回 0，失败返回其他值。

ViewSignerCertWnd　　显示签名者证书窗体

方法声明：

long ViewSignerCertWnd ();

方法说明：

显示签名者证书窗体。前提条件：已经验证签名成功，并且在签名数字信封中包含签名者证书。

注：此函数在 B/S 结构中作为 Web 服务器脚本中不能使用。

参数说明：

返回值：

成功时返回 0，

失败时返回<0，由 GetErrorCode 方法中取错误码，由 GetErrorMessage 方法中取错误信息。

ViewDecryptCertWnd　　显示接收人证书窗体

方法声明：

long ViewDecryptCertWnd ();

方法说明：

显示接收人证书窗体。前提条件：已经解密数字信封成功，并且在加密数字信封中包含接收人证书。

注：此函数在 B/S 结构中作为 Web 服务器脚本中不能使用。

参数说明：

返回值：

成功时返回 0，

失败时返回<0，由 GetErrorCode 方法中取错误码，由 GetErrorMessage 方法中取错误信息。

4.4.3 对称加解密方法

SymCipherInit　　对称加解密初始化

方法声明：

long SymCipherInit(BSTR CipherAlgo, BSTR InKey)

方法说明：

根据指定的加解密算法和口令进行对称加解密初始化。

参数说明：

CipherAlgo: 摘要算法

"1.2.840.113549.3.7" = szOID_RSA_DES_EDE3_CBC,
"1.3.14.3.2.7" = szOID_OIWSEC_desCBC,
"1.2.840.113549.3.4" = szOID_RSA_RC4,
"1.2.840.113549.3.2" = szOID_RSA_RC2CBC

InKey: 口令

返回参数：

成功返回 0，失败返回其他值。

SymEncryptData 对称加密数据

方法声明：

BSTR SymEncryptData(BSTR SourceData)

方法说明：

产生指定原文数据的对称加密密文数据。密文数据为 BASE64 编码格式。前提条件：已经成功地进行对称加解密初始化。

参数说明：

SourceData： 原文数据

其中：通过 InputDataType 属性来指定原文数据格式

0 = 输入原文为二进制编码，此函数内部不进行转码
1 = 输入原文为 BASE64 编码，此函数内部进行转码

返回参数：

成功返回对称加密 BASE64 编码数据，失败返回 NULL。通过 ReturnLong()函数返回对称加密 BASE64 编码数据长度。

SymDecryptData 对称解密数据

方法声明：

BSTR SymDecryptData(BSTR CryptoBase64Data)

方法说明：

对指定的 BASE64 编码格式的对称加密密文数据进行解密操作。前提条件：已经成功地进行对称加解密初始化。

参数说明：

CryptoBase64Data: BASE64 编码格式的对称加密密文数据

返回参数：

通过 OutputDataType 属性来指定返回数据格式

0 = 输出返回数据为二进制编码，此函数内部不进行转码
1 = 输出返回数据为 BASE64 编码，此函数内部进行转码

成功返回原文数据，失败返回 NULL。通过 ReturnLong()函数返回原文数据长度。

SymEncryptFile 对称加密文件

方法声明：

BSTR SymEncryptFile(BSTR SourceFile, BSTR EncFile)

方法说明：

指定原文文件产生对称加密密文数据并保存到指定文件中。前提条件：已经成功地进行对称加解密初始化。

参数说明：

SourceFile： 输入原文文件
EncFile： 输入要保存的密文文件
如果此参数为空，则只返回 BASE64 编码数据

其中：通过 InputDataType 属性来指定原文文件数据格式
0 = 输入原文为二进制编码，此函数内部不进行转码
1 = 输入原文为 BASE64 编码，此函数内部进行转码

通过 OutputDataType 属性来指定返回密文文件数据格式
0 = 输出返回密文文件数据为二进制编码，此函数内部不进行转码
1 = 输出返回密文文件数据为 BASE64 编码，此函数内部进行转码

返回参数：

成功返回对称加密 BASE64 编码数据，失败返回 NULL。通过 ReturnLong()函数返回对称加密 BASE64 编码数据长度。

SymDecryptFile 对称解密文件

方法声明：

BSTR SymDecryptFile(BSTR EncFile, BSTR RetSourceFile)

方法说明：

对指定的密文文件进行对称解密，并将解密结果保存到指定的原文数据文件中。前提条件：已经成功地进行对称加解密初始化。

参数说明：

EncFile: 输入密文文件
RetSourceFile: 输入要保存的原文文件

其中：通过 InputDataType 属性来指定密文文件数据格式
0 = 输入密文为二进制编码，此函数内部不进行转码
1 = 输入密文为 BASE64 编码，此函数内部进行转码

通过 OutputDataType 属性来指定返回解密原文数据格式
0 = 输出返回解密原文数据为二进制编码，此函数内部不进行转码
1 = 输出返回解密原文数据为 BASE64 编码，此函数内部进行转码

返回参数：
成功返回解密原文数据，失败返回 NULL。通过 ReturnLong()函数返回原文数据长度。

4.4.4 摘要方法

HashData 产生数据摘要

方法声明：
BSTR HashData (BSTR sInData, BSTR sHashAlgo)
方法说明：
根据指定的数据摘要算法产生指定原文数据的摘要数据。
前提条件：已经成功初始化引擎。可以通过 StringToByteLength()来获得数据的字节长度。
参数说明：
sInData: 二进制格式的原文数据
sDigestAlgo: 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"

其中：通过 InputDataType 属性来指定原文数据格式
0 = 输入原文为二进制编码，此函数内部不进行转码
1 = 输入原文为 BASE64 编码，此函数内部进行转码

通过 OutputDataType 属性来指定返回的摘要数据格式
0 = 返回的摘要数据格式为 BASE64 编码，此函数内部进行转码
1 = 返回的摘要数据格式为十六进制编码，此函数内部进行转码
返回参数：
成功返回摘要编码数据，失败返回 NULL。通过 ReturnLong()函数返回摘要编码数据长度。

VerifyHashData 验证数据摘要

方法声明：

long VerifyHashData (BSTR sInData,
BSTR sDigestData,
BSTR sHashAlgo)

方法说明：

验证数据摘要，前提条件：已经成功初始化引擎。

参数说明：

sInData: 二进制格式的原文数据
sDigestData: 摘要数据
sDigestAlgo: 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"

其中：通过 InputDataType 属性来指定原文数据格式
0 = 输入原文为二进制编码，此函数内部不进行转码
1 = 输入原文为 BASE64 编码，此函数内部进行转码

通过 OutputDataType 属性来指定摘要数据格式
0 = 摘要数据格式为 BASE64 编码，此函数内部进行转码
1 = 摘要数据格式为十六进制编码，此函数内部进行转码

返回参数：

成功返回 0，失败返回其他值。

HashFile 产生文件摘要

方法声明：

BSTR HashFile (BSTR sInFile, BSTR sHashAlgo)

方法说明：

根据指定的摘要算法产生指定文件的摘要数据。

前提条件：已经成功初始化引擎。可以通过 StringToByteLength()来获得数据的字节长度。

参数说明：

sInFile: 原文文件
sDigestAlgo: 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"

其中：通过 InputDataType 属性来指定原文文件数据格式
0 = 输入原文文件为二进制编码，此函数内部不进行转码
1 = 输入原文文件为 BASE64 编码，此函数内部进行转码

通过 OutputDataType 属性来指定返回的摘要数据格式
0 = 返回的摘要数据格式为 BASE64 编码，此函数内部进行转码
1 = 返回的摘要数据格式为十六进制编码，此函数内部进行转码

返回参数：
成功返回摘要编码数据，失败返回 NULL。通过 ReturnLong()函数返回摘要编码数据长度。

VerifyHashFile 验证文件摘要

方法声明：

```
long VerifyHashFile (BSTR sInFile,
                     BSTR sDigestData,
                     BSTR sHashAlgo)
```

方法说明：
验证文件摘要，前提条件：已经成功初始化引擎。
参数说明：

sInFile:	原文文件
sDigestData:	摘要数据
sDigestAlgo:	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"

其中：通过 InputDataType 属性来指定原文文件数据格式
0 = 输入原文文件为二进制编码，此函数内部不进行转码
1 = 输入原文文件为 BASE64 编码，此函数内部进行转码

通过 OutputDataType 属性来指定摘要数据格式
0 = 摘要数据格式为 BASE64 编码，此函数内部进行转码
1 = 摘要数据格式为十六进制编码，此函数内部进行转码

返回参数：
成功返回 0，失败返回其他值。

4.4.5 数字签名方法

SignHash　产生数字签名

方法声明：

BSTR SignHash (BSTR SourceData,
long nCertIndex,
BSTR sHashAlgo,
BSTR pPin)

方法说明：

根据指定的包含私钥的签名者证书和签名算法，产生指定原文数据的数字签名数据。数字签名数据为 BASE64 编码格式。前提条件：已经成功初始化引擎。

参数说明：

SourceData：　原文数据

long nCertIndex：　指定使用的签名证书编号

sHashAlgo：　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

其中：通过 InputDataType 属性来指定原文数据格式

0 = 输入原文为二进制编码，此函数内部不进行转码

1 = 输入原文为 BASE64 编码，此函数内部进行转码

返回参数：

成功返回签名 BASE64 编码数据，失败返回 NULL。通过 ReturnLong()函数返回签名 BASE64 编码数据长度。

VerifySignHash　验证数字签名

方法声明：

long VerifySignHash (BSTR SourceData,
BSTR SignHashBase64Data,
long nCertIndex,
BSTR sHashAlgo)

方法说明：

根据指定的原文数据、签名者公钥证书和签名算法，验证指定的数字签名数据的一致性和有效性。前提条件：已经成功初始化引擎。

参数说明：

SourceData：　　原文数据
SignHashBase64Data：　数字签名数据(BASE64 编码)
long nCertIndex：　指定使用验证签名公钥证书编号
sHashAlgo：　　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”

其中：通过 InputDataType 属性来指定原文数据格式
　　0 = 输入原文为二进制编码，此函数内部不进行转码
　　1 = 输入原文为 BASE64 编码，此函数内部进行转码

返回参数：
　　成功返回 0；失败返回<0，由 GetErrorCode 方法中取错误码，由 GetErrorMessage 方法中取错误信息。

SignHashFile　产生数字签名文件

方法声明：

```
BSTR SignHashFile (BSTR SourceFile,
                   long nCertIndex,
                   BSTR sHashAlgo,
                   BSTR pPin)
```

方法说明：
　　根据指定的包含私钥的签名者证书和签名算法，比较指定原文文件的数字签名数据。数字签名数据为 BASE64 编码格式。前提条件：已经成功初始化引擎。
参数说明：
SourceFile：原文文件
long nCertIndex：　指定使用的签名证书编号
sHashAlgo：　　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

其中：通过 InputDataType 属性来指定原文文件数据格式
　　0 = 输入原文为二进制编码，此函数内部不进行转码
　　1 = 输入原文为 BASE64 编码，此函数内部进行转码

返回参数：
　　成功返回签名 BASE64 编码数据，失败返回 NULL。通过 ReturnLong()函数返回签名 BASE64 编码数据长度。

VerifySignHashFile 验证数字签名文件

方法声明：

long VerifySignHashFile (BSTR SourceFile,
BSTR SignHashBase64Data,
long nCertIndex,
BSTR sHashAlgo)

方法说明：

根据指定的原文文件、签名者公钥证书和签名算法，验证指定的数字签名数据的一致性和有效性。前提条件：已经成功初始化引擎。

参数说明：

SourceFile： 原文文件
SignHashBase64Data： 数字签名数据(BASE64 编码)
long nCertIndex： 指定使用验证签名公钥证书编号
sHashAlgo： 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”

其中：通过 InputDataType 属性来指定原文文件数据格式
0 = 输入原文为二进制编码，此函数内部不进行转码
1 = 输入原文为 BASE64 编码，此函数内部进行转码

返回参数：

成功返回 0；失败返回<0，由 GetErrorCode 方法中取错误码，由 GetErrorMessage 方法中取错误信息。

SignFromHash 产生数字签名

方法声明：

BSTR SignFromHash (BSTR HashData,
long nCertIndex,
BSTR sHashAlgo,
BSTR pPin)

方法说明：

根据指定的包含私钥的签名者证书和签名算法，产生指定哈希数据的数字签名数据。数字签名数据为 BASE64 编码格式。前提条件：已经成功初始化引擎。

参数说明：

HashData： 哈希数据
long nCertIndex： 指定使用的签名证书编号
sHashAlgo： 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

其中：通过 InputDataType 属性来指定哈希数据格式

0 = 输入哈希为二进制编码，此函数内部不进行转码

1 = 输入哈希为 BASE64 编码，此函数内部进行转码

返回参数：

成功返回签名 BASE64 编码数据，失败返回 NULL。通过 ReturnLong()函数返回签名 BASE64 编码数据长度。

VerifySignFromHash　验证数字签名

方法声明：

long VerifySignFromHash (BSTR HashData,
BSTR SignHashBase64Data,
long nCertIndex,
BSTR sHashAlgo)

方法说明：

根据指定的哈希数据、签名者公钥证书和签名算法，验证指定的数字签名数据的一致性和有效性。前提条件：已经成功初始化引擎。

参数说明：

HashData：　　哈希数据

SignHashBase64Data：　数字签名数据(BASE64 编码)

long nCertIndex：　指定使用验证签名公钥证书编号

sHashAlgo：　　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”

其中：通过 InputDataType 属性来指定哈希数据格式

0 = 输入哈希为二进制编码，此函数内部不进行转码

1 = 输入哈希为 BASE64 编码，此函数内部进行转码

返回参数：

成功返回 0；失败返回<0，由 GetErrorCode 方法中取错误码，由 GetErrorMessage 方法中取错误信息。

4.4.6 签名数字信封方法

SignInit　　签名数字信封初始化

方法声明：

long SignInit (long SignerCertIndex,
BSTR SignAlgo,
long bIsAddSignerCert,
long bIsAddSrc,
long IsAddTime,
BSTR SignDataType,
BSTR pPin)

方法说明：

根据指定的包含私钥的签名者签名证书以及相关参数，进行签名数字信封初始化，前提条件：已经成功初始化引擎。只有进行初始化后，才能进行下一步的签名操作。

参数说明：

SignerCertIndex　　指定使用包含私钥的签名者签名证书编号

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"

bIsAddSignerCert:　是否在结果中携带证书

0 = 不携带证书
1 = 携带证书

bIsAddSrc:　　是否在结果中携带原文

0 = 不携带原文
1 = 携带原文

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。请参考时间戳操作。

SignDataType:　　签名数据类型(通常:NULL)

szOID_TSP_TSTInfo = "1.2.840.113549.1.9.16.1.4"

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

返回参数：

成功返回 0，失败返回<0，由 GetErrorCode 方法中取错误码，由 GetErrorMessage 方法中取错误信息。

SignData　产生签名数字信封

方法声明：

BSTR SignData (BSTR SourceData)

方法说明：

产生指定原文数据的签名数字信封数据。数字信封数据为 BASE64 编码格式。

前提条件：已经成功初始化引擎。已经成功地进行签名数据初始化。

参数说明：

SourceData：　　原文数据

其中：通过 InputDataType 属性来指定原文数据格式

0 = 输入原文为二进制编码，此函数内部不进行转码

1 = 输入原文为 BASE64 编码，此函数内部进行转码

返回参数：

成功返回签名数据 BASE64 编码数据，失败返回 NULL。通过 ReturnLong()函数返回签名数据 BASE64 编码数据长度。

VerifySign　验证签名数据

方法声明：

BSTR VerifySign (BSTR SourceData ,
BSTR Base64EncodedMessage,
long SignerCertIndex)

方法说明：

根据指定的签名者签名公钥证书，验证指定的签名数字信封数据，产生原文数据。

前提条件：已经成功初始化引擎。

参数说明：

SourceData：　原文数据，如果为空，则使用签名数据中包含的原文。当本参数为空，同时签名数据中又没有包含原文数据时，验证签名数据失败。

Base64EncodedMessage：　签名数据(BASE64 编码)

SignerCertIndex：　指定使用签名公钥证书编号

-1 = 一个特例，即不用指定证书，签名数据内包含签名者证书

其中：通过 InputDataType 属性来指定原文数据格式

0 = 输入原文数据为二进制编码，此函数内部不进行转码

1 = 输入原文数据为 BASE64 编码，此函数内部进行转码

通过 OutputDataType 属性来指定返回数据格式

0 = 输出返回数据为二进制编码，此函数内部不进行转码

1 = 输出返回数据为 BASE64 编码，此函数内部进行转码

返回参数：

成功返回原文数据，失败返回 NULL。通过 ReturnLong()函数返回原文数据长度。

SignFile　　产生签名数字信封文件

方法声明：

BSTR SignFile (BSTR SourceFile, BSTR SignFile)

方法说明：

对文件进行签名，产生包含原文、包含签名者证书的签名数字信封文件。

前提条件：已经成功初始化引擎。已经成功地进行签名数字信封初始化。

参数说明：

SourceFile:　　指定原文文件(全路径)

SignFile:　　指定要保存的签名数据文件(全路径)

如果 SignFile 参数为空，只从返回值返回签名的 BASE64 编码数据。

其中：通过 InputDataType 属性来指定原文数据文件格式

0 = 输入原文数据为二进制编码，此函数内部不进行转码

1 = 输入原文数据为 BASE64 编码，此函数内部进行转码

通过 OutputDataType 属性来指定签名数据文件格式

0 = 输出返回数据为二进制编码，此函数内部不进行转码

1 = 输出返回数据为 BASE64 编码，此函数内部进行转码

返回参数：

成功时，返回签名的 BASE64 编码数据。

失败时返回空，由 GetErrorCode 方法中取错误码，由 GetErrorMessage 方法中取错误信息。通过 ReturnLong()函数返回签名数据 BASE64 编码数据长度。

VerifySignFile　　验证签名文件

方法声明：

BSTR VerifySignFile (BSTR szEvpFile, BSTR szSrcFile, long nSignCertIndex);

方法说明：

验证指定的签名数字信封文件，签名数字信封中包含原文、二进制编码签名数据并将原文结果保存到文件。

参数说明：

szEvpFile：　输入签名数据文件名

szSrcFile：　输出原文数据的文件名

如果 szSrcFile 参数为空，只从返回值返回原文结果数据。

nCertIndex：　指定使用的验证签名公钥证书

-1 = 一个特例，即不用指定证书，签名数据内包含签名者证书

其中：通过 InputDataType 属性来指定签名数字信封文件格式

0 = 输入签名数字信封为二进制编码，此函数内部不进行转码
1 = 输入签名数字信封为 BASE64 编码，此函数内部进行转码

通过 OutputDataType 属性来指定返回原文结果文件格式
0 = 输出返回原文结果为二进制编码，此函数内部不进行转码
1 = 输出返回原文结果为 BASE64 编码，此函数内部进行转码

返回值：
成功时，返回原文数据。
失败时返回空，由 GetErrorCode 方法中取错误码，由 GetErrorMessage 方法中取错误信息。

VerifySignDataToFile 验证签名数据-文件

方法声明：

```
BSTR VerifySignDataToFile (BSTR Base64EncodedMessage,
                           BSTR szSrcFile,
                           long nSignCertIndex);
```

方法说明：
验证签名数据-文件操作，验证由签名数据中包含原文、BASE64 编码签名数据并将原文结果保存到文件。

参数说明：

Base64EncodedMessage： 输入签名数据文件名(BASE64 编码)

szSrcFile： 输出原文数据的文件名

通过 OutputDataType 属性来指定返回原文结果文件格式
0 = 输出返回原文结果为二进制编码，此函数内部不进行转码
1 = 输出返回原文结果为 BASE64 编码，此函数内部进行转码

nCertIndex： 指定使用的验证签名公钥证书
-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 GetErrorCode 方法中取错误码，由 GetErrorMessage 方法中取错误信息。

ReturnSignerCertData 获得签名者公钥证书编码数据

方法声明：

```
BSTR ReturnSignerCertData ();
```

方法说明：
返回签名数字信封中包含的签名者公钥证书 BASE64 编码数据，前提条件：已经成功地验证签名数字信封。

注意：如果签名数字信封中不包含签名者证书，则无法返回证书数据。

参数说明：

返回值：

成功时返回 BASE64 编码证书数据，

失败时返回空，由 GetErrorCode 方法中取错误码，由 GetErrorMessage 方法中取错误信息。

ReturnSignerCertSN　获得签名者证书序列号

方法声明：

BSTR ReturnSignerCertSN (BSTR Base64SignedMessage);

方法说明：

根据 BASE64 编码签名数字信封返回签名者证书序列号。

参数说明：

Base64SignedMessage:　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 GetErrorCode 方法中取错误码，由 GetErrorMessage 方法中取错误信息。

ReturnSigningTime　获得签名时间

方法声明：

BSTR ReturnSigningTime (BSTR Base64SignedMessage);

方法说明：

根据 BASE64 编码签名数字信封返回签名时间。(格式 YYYY-MM-DD hh:mm:ss)。

注意：如果签名数字信封时不包含签名时间，则无法返回签名时间。

参数说明：

Base64SignedMessage:　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 GetErrorCode 方法中取错误码，由 GetErrorMessage 方法中取错误信息。

4.4.7 加密数字信封方法

EncryptInit　RSA 加密数字信封初始化

方法声明：

long EncryptInit (long EncCertIndex, BSTR RSAEncAlgo)

方法说明：

根据指定的接收者加密公钥证书对象和加密算法，进行 RSA 加密数字信封初始化，前提条件：已经成功初始化引擎。只有进行初始化后，才能进行下一步的 RSA 加密操作。

参数说明：

EncCertIndex: 指定使用加密证书编号

RSAEncAlgo: RSA 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"

返回参数：

成功返回 0，失败返回<0，由 GetErrorCode 方法中取错误码，由 GetErrorMessage 方法中取错误信息。

EncryptInitAppend 追加 RSA 加密数字信封初始化

方法声明：

long EncryptInitAppend (long EncCertIndex)

方法说明：

根据指定的接收人加密证书，追加 RSA 加密数字信封初始化，此函数用于多证书加密时使用。

前提条件：已经成功初始化引擎。只有进行 EncryptInit ()初始化后，才能进行下一步的 RSA 加密操作。

参数说明：

EncCertIndex 指定使用加密证书编号

返回参数：

成功返回 0，失败返回<0，由 GetErrorCode 方法中取错误码，由 GetErrorMessage 方法中取错误信息。

EncryptData RSA 加密数据

方法声明：

BSTR EncryptData (BSTR SourceData)

方法说明：

产生指定的原文数据的加密数字信封数据。数字信封数据为 BASE64 编码格式。

前提条件：已经成功初始化引擎。已经成功地进行 RSA 加密数据初始化。

参数说明：

SourceData： 原文数据

其中：通过 InputDataType 属性来指定原文数据格式

0 = 输入原文为二进制编码，此函数内部不进行转码

1 = 输入原文为 BASE64 编码，此函数内部进行转码

返回参数：

成功时，返回 RSA 加密的 BASE64 编码数据，

失败返回 NULL。通过 ReturnLong()函数返回 RSA 加密数据 BASE64 编码数据长度。

DecryptData　RSA 解密数据

方法声明：

BSTR DecryptData (BSTR Base64EncodedMessage, long nCertIndex)

方法说明：

解密指定的加密数字信封数据，返回解密原文数据。前提条件：已经成功初始化引擎。

参数说明：

Base64EncodedMessage:　RSA 加密数据(BASE64 编码)

nCertIndex:　指定包含私钥的证书索引号，此索引号应 >= 4

通过 OutputDataType 属性来指定返回数据格式

0 = 输出返回数据为二进制编码，此函数内部不进行转码

1 = 输出返回数据为 BASE64 编码，此函数内部进行转码

返回参数：

成功返回解密原文数据，失败返回 NULL。通过 ReturnLong()函数返回解密原文数据长度。

EncryptFile　RSA 加密文件

方法声明：

BSTR EncryptFile (BSTR SourceFile, BSTR EncFile)

方法说明：

对文件进行 RSA 加密，产生包含原文、包含接收人证书的 RSA 加密数据文件。

前提条件：已经成功初始化引擎。已经成功地进行 RSA 加密数据初始化。

参数说明：

SourceFile:　指定原文文件(全路径)

SignFile:　指定要保存的 RSA 加密文件(全路径)

其中：通过 InputDataType 属性来指定原文数据文件格式

0 = 输入原文数据为二进制编码，此函数内部不进行转码

1 = 输入原文数据为 BASE64 编码，此函数内部进行转码

通过 OutputDataType 属性来指定 RSA 加密数据文件格式

0 = 输出返回数据为二进制编码，此函数内部不进行转码

1 = 输出返回数据为 BASE64 编码，此函数内部进行转码

返回参数：

成功时，返回 RSA 加密的 BASE64 编码数据。

失败时返回空，由 GetErrorCode 方法中取错误码，由 GetErrorMessage 方法中取错误信息。通过 ReturnLong()函数返回 RSA 加密数据 BASE64 编码数据长度。

DecryptFile　　RSA 解密文件

方法声明：

BSTR DecryptFile (BSTR EncFile, BSTR SourceFile, long nIndexCert);

方法说明：

解密指定加密数字信封文件，返回解密数据。前提条件：已经成功初始化引擎。

RSA 数据文件操作，验证由 EncryptData 生成的 RSA 加密文件。

参数说明：

EncFile：　　输入 RSA 加密数据文件名

SourceFile：　输出解密原文数据的文件名

nIndexCert:　指定要解密的证书索引号

其中：通过 InputDataType 属性来指定加密数字信封文件格式

0 = 输入加密数字信封为二进制编码，此函数内部不进行转码

1 = 输入加密数字信封为 BASE64 编码，此函数内部进行转码

通过 OutputDataType 属性来指定返回解密原文结果文件格式

0 = 输出返回解密原文结果为二进制编码，此函数内部不进行转码

1 = 输出返回解密原文结果为 BASE64 编码，此函数内部进行转码

返回值：

成功时，返回解密原文数据。

失败时返回空，由 GetErrorCode 方法中取错误码，由 GetErrorMessage 方法中取错误信息。

DecryptDataToFile　　RSA 解密数字信封数据-文件

方法声明：

BSTR DecryptDataToFile(BSTR Base64EncodedMessage, BSTR SourceFile, long nIndexCert);

方法说明：

RSA 解密数字信封数据-文件操作，解密由 RSA 加密数据中包含原文、BASE64 编码 RSA 加密数据并将原文结果保存到文件。

参数说明：

Base64EncodedMessage：　输入 RSA 加密数据文件名(BASE64 编码)

SourceFile：　　输出原文数据的文件名

通过 OutputDataType 属性来指定返回解密原文结果文件格式

0 = 输出返回解密原文结果为二进制编码，此函数内部不进行转码

1 = 输出返回解密原文结果为 BASE64 编码，此函数内部进行转码

nIndexCert: 指定要解密的证书索引号

返回值：

成功时，返回原文数据。

失败时返回空，由 GetErrorCode 方法中取错误码，由 GetErrorMessage 方法中取错误信息。

ReturnEncerCertData 获得接收人公钥证书编码数据

方法声明：

BSTR ReturnEncerCertData ();

方法说明：

返回加密数字信封中包含的接收人公钥证书 BASE64 编码数据，前提条件：已经成功地解密数字信封。

注意：如果加密数字信封中不包含接收人证书，则无法返回证书数据。

参数说明：

返回值：

成功时返回 BASE64 编码证书数据，

失败时返回空，由 GetErrorCode 方法中取错误码，由 GetErrorMessage 方法中取错误信息。

4.4.8 签名加密数字信封方法

SignEnvData 产生签名加密数字信封

方法声明：

BSTR SignEnvData (BSTR SourceData)

方法说明：

根据指定的包含私钥的发送者签名证书和接收者加密公钥证书，产生指定原文数据的签名加密数字信封数据。数字信封数据为 BASE64 编码格式。

前提条件：已经成功初始化引擎。已经成功地进行签名数据初始化和 RSA 加密数据初始化。

参数说明：

SourceData: 原文数据

其中：通过 InputDataType 属性来指定原文数据格式

0 = 输入原文为二进制编码，此函数内部不进行转码

1 = 输入原文为 BASE64 编码，此函数内部进行转码

返回参数：

成功返回签名加密数字信封 BASE64 编码数据，

失败返回 NULL。通过 ReturnLong()函数返回签名加密数字信封 BASE64 编码数据

长度。

VfyDecData　验证并拆解签名加密数字信封数据

方法声明：

BSTR VfyDecData (BSTR Base64EncodedMessage,
long SignerCertIndex,
long EncCertIndex)

方法说明：

解密并验证指定的签名加密数字信封数据，产生二进制格式原文数据。前提条件：已经成功初始化引擎。

参数说明：

Base64EncodedMessage:　BASE64 格式的签名加密数字信封数据
SignerCertIndex:　指定使用签名公钥证书编号
-1 = 一个特例，即不用指定证书，签名数据内包含签名者证书
EncCertIndex:　指定要解密的证书索引号

通过 OutputDataType 属性来指定返回数据格式
0 = 输出返回数据为二进制编码，此函数内部不进行转码
1 = 输出返回数据为 BASE64 编码，此函数内部进行转码

返回参数：

成功返回原文数据，失败返回 NULL。通过 ReturnLong()函数返回原文数据长度。

SignEncFile　产生签名加密数字信封文件

方法声明：

BSTR SignEncFile (BSTR SourceFile, BSTR SignEncFile)

方法说明：

对文件进行签名加密，产生包含原文、发送者签名证书、接收人证书的签名加密数字信封文件。

前提条件：已经成功初始化引擎。已经成功地进行签名初始化和 RSA 加密数据初始化。

参数说明：

SourceFile:　指定原文文件(全路径)
SignEncFile:　指定要保存的签名加密数字信封文件(全路径)

其中：通过 InputDataType 属性来指定原文数据文件格式
0 = 输入原文数据为二进制编码，此函数内部不进行转码
1 = 输入原文数据为 BASE64 编码，此函数内部进行转码

通过 OutputDataType 属性来指定 RSA 加密数据文件格式
0 = 输出返回数据为二进制编码，此函数内部不进行转码

1 = 输出返回数据为 BASE64 编码，此函数内部进行转码

返回参数：

成功时，返回签名 RSA 加密的 BASE64 编码数据。通过 ReturnLong()函数返回签名加密数字信封数据 BASE64 编码数据长度。

失败时返回空，由 GetErrorCode 方法中取错误码，由 GetErrorMessage 方法中取错误信息。

VfyDecFile 验证并拆解签名加密数字信封文件

方法声明：

```
BSTR VfyDecFile (BSTR EncFile,
                 BSTR SourceFile,
                 long SignerCertIndex,
                 long EncCertIndex);
```

方法说明：

解密并验证指定的签名加密数字信封数据，产生原文文件。前提条件：已经成功初始化引擎。

参数说明：

EncFile：输入签名加密数字信封文件名

SourceFile：要输出原文数据的文件名

SignerCertIndex:指定使用签名公钥证书编号

-1 = 一个特例，即不用指定证书，签名数据内包含签名者证书

EncCertIndex:指定要解密的证书索引号

其中：通过 InputDataType 属性来指定签名加密数字信封文件格式

0 = 输入签名加密数字信封为二进制编码，此函数内部不进行转码

1 = 输入签名加密数字信封为 BASE64 编码，此函数内部进行转码

通过 OutputDataType 属性来指定返回原文结果文件格式

0 = 输出返回原文结果为二进制编码，此函数内部不进行转码

1 = 输出返回原文结果为 BASE64 编码，此函数内部进行转码

返回值：

成功时，返回原文数据。

失败时返回空，由 GetErrorCode 方法中取错误码，由 GetErrorMessage 方法中取错误信息。

VfyDecDataToFile 验证并拆解签名加密数字信封数据-文件

方法声明：

```
BSTR VfyDecDataToFile (BSTR Base64EncodedMessage,
                       BSTR SourceFile,
```

long SignerCertIndex,
long EncCertIndex);

方法说明：

解密并验证指定的签名加密数字信封数据，产生原文文件。前提条件：已经成功初始化引擎。

参数说明：

Base64EncodedMessage：　输入 BASE64 编码签名加密数字信封文件名

SourceFile：　输出原文数据的文件名

通过 OutputDataType 属性来指定返回解密原文结果文件格式

0 = 输出返回解密原文结果为二进制编码，此函数内部不进行转码

1 = 输出返回解密原文结果为 BASE64 编码，此函数内部进行转码

SignerCertIndex:　指定使用签名公钥证书编号

-1 = 一个特例，即不用指定证书，签名数据内包含签名者证书

EncCertIndex:　指定要解密的证书索引号

返回值：

成功时，返回原文数据。

失败时返回空，由 GetErrorCode 方法中取错误码，由 GetErrorMessage 方法中取错误信息。

4.4.9 时间戳方法

SetTimeStampServerUrl　设置时间戳服务器地址

方法声明：

long SetTimeStampServerUrl(BSTR sUrl);

方法说明：

向 COM 引擎对象中设置时间戳服务器地址。

参数说明：

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 追加时间戳服务器地址

方法声明：

long AppendTimeStampServerUrl(BSTR sUrl);

方法说明：

向 COM 引擎对象中追加时间戳服务器地址。

参数说明：

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 发送时间戳请求

方法声明：

long RequestTimeStamp (BSTR sData, BSTR szHashAlgorithmOID);

方法说明：

发送时间戳请求，同时返回时间戳响应。前提条件：已经设置或追加时间戳服务器地址。

参数说明：

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 GetErrorCode 方法中取错误码，由 GetErrorMessage 方法中取错误信息。

GetTimeStampVersion 获得时间戳版本

方法声明：

BSTR GetTimeStampVersion();

方法说明：

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。

参数说明：

返回值：

成功时返回时间戳版本，

失败时返回空，由 GetErrorCode 方法中取错误码，由 GetErrorMessage 方法中取错误信息。

GetTimeStampTimeStr 获得时间戳签名时间字符串

方法声明：

BSTR GetTimeStampTimeStr ();

方法说明：

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。

参数说明：

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 GetErrorCode 方法中取错误码，由 GetErrorMessage 方法中取错误信息。

GetTimeStampSerialNumber 获得时间戳签名证书序列号

方法声明：

BSTR GetTimeStampSerialNumber ();

方法说明：

获得时间戳的序列号。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。

参数说明：

返回值：

成功时返回时间戳的序列号，

失败时返回空，由 GetErrorCode 方法中取错误码，由 GetErrorMessage 方法中取错误信息。

GetTimeStampMessageImprint 获得时间戳签名消息拇印

方法声明：

BSTR GetTimeStampMessageImprint ();

方法说明：

获得时间戳签名消息拇印。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。

参数说明：

返回值：

成功时返回时间戳签名消息拇印，

失败时返回空，由 GetErrorCode 方法中取错误码，由 GetErrorMessage 方法中取错误信息。

GetTimeStampToken 获得时间戳响应

方法声明：

BSTR GetTimeStampToken ();

方法说明：

获得时间戳响应信息。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。

参数说明：

返回值：

成功时返回时间戳响应信息，

失败时返回空，由 GetErrorCode 方法中取错误码，由 GetErrorMessage 方法中取错

误信息。

GetTimeStampSrcHash 获得时间戳中原文数据哈希值

函数声明：

BSTR GetTimeStampSrcHash ();

方法说明：

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。

参数说明：

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 GetErrorCode 方法中取错误码，由 GetErrorMessage 方法中取错误信息。

GetTimeStampSrcAlgo 获得时间戳中包含的算法

函数声明：

BSTR GetTimeStampSrcAlgo ();

方法说明：

获得时间戳中包含的原文数据哈希算法。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。

参数说明：

返回值：

成功时返回时间戳中包含的原文数据哈希算法，

失败时返回空，由 GetErrorCode 方法中取错误码，由 GetErrorMessage 方法中取错误信息。

ViewTimeStampCertWnd 显示时间戳签名证书窗体

函数声明：

long ViewTimeStampCertWnd ();

方法说明：

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。

注：此函数在 B/S 结构中作为 Web 服务器脚本中不能使用。

参数说明：

返回值：

成功时返回 0，

失败时返回<0，由 GetErrorCode 方法中取错误码，由 GetErrorMessage 方法中取错误信息。

4.4.10 其他公共方法

GetVersion

函数声明：

BSTR GetVersion ();

方法说明：

获得该 COM 组件的版本，通过外部脚本，可以用于判断 COM 组件是否为最新版本，即是否可以使用。

参数说明：

返回值：

返回获得该 COM 组件的版本

GetErrorCode 获取错误代码

函数声明：

long ErrorCode();

方法说明：

获取错误号，在程序出错的时候，调用本方法。

参数说明：

返回值：

0 表示成功；<0 值为最后发生的错误号码。

GetErrorMessage 获取错误信息

函数声明：

BSTR GetErrorMessage ();

方法说明：

获取错误信息，在程序出错的时候，调用本方法。

参数说明：

返回值：

BinToBase64 二进制数据转成 BASE64 编码数据

函数声明：

BSTR BinToBase64 (BSTR sSource, long SourceLen, long IsCert);

方法说明：

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。

参数说明：

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 [可选参数]，默认：0

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()方法中取错误码，由 GetErrorMessage()方法中取错误信息。

Base64ToBin　BASE64 编码数据转成二进制数据

函数声明：

BSTR Base64ToBin (BSTR sEncoded, long EncodedLen, long IsCert);

方法说明：

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。

参数说明：

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 [可选参数]，默认：0

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()方法中取错误码，由 GetErrorMessage()方法中取错误信息。

BinToBase64File　二进制编码文件转成 BASE64 编码文件

函数声明：

long BinToBase64File (BSTR sBinFile, BSTR sBase64File);

方法说明：

将二进制编码文件转成 BASE64 编码文件。

参数：

sBinFile：　二进制编码文件(全路径)

sBase64File：BASE64 编码文件(全路径)　[可选参数]，默认：””

如果 sBase64File 参数为空，则将产生的 BASE64 编码数据写入 sBinFile 文件中。

返回值：

成功时返回 0，

失败时返回<0，由 ErrorCode()方法中取错误码，由 GetErrorMessage()方法中取错误信息。

Base64ToBinFile　BASE64 编码文件转成二进制编码文件

函数声明：

long Base64ToBinFile (BSTR sBase64File, BSTR sBinFile);

方法说明：

将 BASE64 编码文件转成二进制编码文件。

参数说明：

sBase64File：BASE64 编码文件(全路径)

sBinFile：　二进制编码文件(全路径)　[可选参数]，默认：””

如果 sBinFile 参数为空，则将产生的二进制编码数据写入 sBase64File 文件中。

返回值：

成功时返回 0，

失败时返回<0，由 ErrorCode()方法中取错误码，由 GetErrorMessage()方法中取错误信息。

GetStdTime　从标准时间服务器获得时间串

函数声明：

BSTR GetStdTime (BSTR sUrl, long nPort);

方法说明：

从标准时间服务器获得时间串。

返回时间字符串(格式：YYYY-MM-DD hh:mm:ss.ms)

参数说明：

BSTR sUrl　　标准时间服务器地址，可选参数，默认：time.windows.com

long　nPort　　标准时间服务器端口，可选参数，默认：123

返回值：

成功时返回时间串，

失败时返回空，由 ErrorCode()方法中取错误码，由 GetErrorMessage()方法中取错误信息。

SetStdTime　从标准时间服务器设置时间串

函数声明：

BSTR SetStdTime (BSTR sUrl, long nPort);

方法说明：

从标准时间服务器获得时间，并设置本 Com 组件所在的系统时间。

返回时间字符串(格式：YYYY-MM-DD hh:mm:ss.ms)

参数说明：

BSTR sUrl　　标准时间服务器地址，可选参数，默认：time.windows.com

long　nPort　　标准时间服务器端口，可选参数，默认：123

返回值：

成功时返回时间串，

失败时返回空，由 ErrorCode()方法中取错误码，由 GetErrorMessage()方法中取错误信息。

SystemBeginTime　　获得 Com 组件系统时间串并设置起始时间

函数声明：

BSTR SystemBeginTime (long nTimeFlag);

方法说明：

获得 Com 组件系统时间串，同时保存此时间为起始时间。Com 组件定义时自动调用此函数，因此此函数不是必须调用的。

返回时间字符串(格式：YYYY-MM-DD hh:mm:ss.ms)

参数说明：

nTimeFlag：时间标志

0 = 获得起始时间

1 = 重新设置起始时间，并获得起始时间

返回值：

成功时返回时间串，

失败时返回空，由 ErrorCode()方法中取错误码，由 GetErrorMessage()方法中取错误信息。

SystemEndTime　　获得 Com 组件系统时间串并计算时间长度

函数声明：

BSTR SystemEndTime (long nTimeUnitType);

方法说明：

获得 Com 组件系统时间串，同时计算此时间与起始时间之差值。

返回时间字符串(格式：YYYY-MM-DD hh:mm:ss.ms)

参数说明：

nTimeUnitType:　返回时间单位类型

1 = 毫秒

2 = 秒

3 = 分

4 = 小时

5 = 天

返回值：

成功时返回时间串，通过 ReturnLong()函数获得终止时间与起始时间之差值。

失败时返回空，由 ErrorCode()方法中取错误码，由 GetErrorMessage()方法中取错误信息。

ReturnLong　获得长度

函数声明：

long ReturnLong ();

方法说明：

读取数据长度，前提条件：成功地调用了 Base64ToBin()函数

参数说明：

返回值：

返回数据长度

StringToByteLength 获得字符串的 Ansi 字节长度

函数声明：

long StringToByteLength(BSTR Str);

方法说明：

获得字符串的 Ansi 字节长度

参数说明：

Str：原文数据

返回值：

成功返回原文数据长度，失败返回 0

StringToUnicodeLength 获得字符串的 Unicode 字节长度

函数声明：

long StringToUnicodeLength (BSTR Str);

方法说明：

获得字符串的 Unicode 字节长度

参数说明：

Str：原文数据

返回值：

成功返回原文数据长度，失败返回 0

GenUUID　产生 UUID 值

函数声明：

BSTR GenUUID (long iRetType);

方法说明：

产生 UUID。UUID = universally unique identifier.

参数说明：

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()方法中取错误码，由 GetErrorMessage()方法中取错误信息。

CryptGenRandom 产生随机数

函数声明：

BSTR CryptGenRandom (long iRandLen, long RetType);

方法说明：

产生指定长度的随机数。

参数说明：

iRandLen：指定要产生的随机数长度

RetType： 返回数据编码类型 [可选参数]，默认：0

0: BASE64 编码, 1: 十六进制编码, 2: 二进制编码

产生随机数长度:

BASE64 编码 = dwRandLen*4/3

十六进制编码 = dwRandLen*2

二进制编码 = dwRandLen

返回值：

成功时返回 BASE64 编码随机数，

失败时返回空，由 ErrorCode()方法中取错误码，由 GetErrorMessage()方法中取错误信息。

ReadFromLocalFile 读取本地磁盘文件

函数声明：

BSTR ReadFromLocalFile (BSTR InputFile);

方法说明：

从本地磁盘文件读取数据。

参数说明：

InputFile： 指定要读取的文件名(全路径)

其中：通过 OutputDataType 属性来指定返回文件数据格式

0 = 输出返回文件数据结果为二进制编码，此函数内部不进行转码

1 = 输出返回文件数据结果为 BASE64 编码，此函数内部进行转码

返回值：

成功时返回文件中数据，

失败时返回空，由 ErrorCode()方法中取错误码，由 GetErrorMessage()方法中取错误信息。

WriteToLocalFile 数据写入本地磁盘文件

函数声明：

long WriteToLocalFile (BSTR InputFile,
BSTR szSrcData,
long SrcDataLen);

方法说明：

将数据写入本地磁盘文件。

参数说明：

InputFile： 指定要写入的文件名(全路径)
szSrcData： 要写入的数据
SrcDataLen： 要写入的数据长度

其中：通过 InputDataType 属性来指定原文数据文件格式
0 = 输入原文数据为二进制编码，此函数内部不进行转码
1 = 输入原文数据为 BASE64 编码，此函数内部进行转码

返回值：

成功时返回 0，
失败时返回<0，由 ErrorCode()方法中取错误码，由 GetErrorMessage()方法中取错误信息。

4.4.11 ASP 中调用引擎接口

```
Dim objEngine
Set objEngine = Server.CreateObject("LNCACryptoCom.LNCAEngineCom.1")
Ret = objEngine.CreateEngine( CStr(strWorkPath) )
nErrorCode = objEngine.ErrorCode
If 0 <> nErrorCode Then
    Response.Write "初始化引擎对象失败！ " & "ErrorCode = " & nErrorCode & "<br>"
    Response.End()
Else
    Response.Write("初始化引擎对象成功.<br>")
    Set Session("obj_Engine") = objEngine
End If
```

4.5 LNCACertificateCom 接口

位于 Com 库的 LNCACertificateCom 接口用于定义封装数字证书数据的实现类或硬件设备中证书对象代理的实现类应提供的功能。可以从一个引擎接口对象中导出并初始化证书接口对象，或者从一个证书文件初始化证书接口对象，也可以从 LADP 服务器下载

并初始化证书接口对象，在证书接口对象中包含一个符合 X.509 标准的公钥证书对象，它们代表辽宁省数字证书认证中心颁发的数字证书。

通常，为了保证证书私钥的安全性，个人证书和企业证书采用 USB-KEY 作为其存储介质，而服务器证书采用加密机或加密卡作为存储介质。

证书接口对象主要功能：证书解析、证书下载、证书验证等。

4.5.1 证书相关属性

ShowErrorMode 显示错误模式

属性声明：

获得：long ShowErrorMode ()

设置：ShowErrorMode (long newVal)

属性说明：

获得或设置显示错误模式，该接口属性为在调试时使用而设置的。

默认时，可以不调用此属性。

注：此属性在 B/S 结构中作为 Web 服务器脚本中不能使用。

参数说明：

lValue：状态值

0 = 通过 ErrorCode()方法返回错误代码或者 GetErrorMessage()方法返回错误信息

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

RDNType 证书主题显示格式类型

属性声明：

获得：long RDNType()

设置：RDNType(long newVal)

属性说明：

获得或者设置证书颁发者和主题显示格式类型，此项设置是在"获得证书主题 GetSubject()、获得证书颁发者 GetIssuer()"时起作用。

默认时，组件自动使用 1 类型，可以不调用此函数。

参数说明：

newVal: RDN 的格式

0 = 微软类型: 以"S="描述省份， 每个子项以"逗号+空格"间隔

1 = RFC 类型: 以"ST="描述省份，每个子项以"逗号"间隔

4.5.2 证书初始化相关方法

SetEngineInfo　　设置引擎信息

方法声明：

long SetEngineInfo (LPDISPATCH I_Engine);

方法说明：

通过引擎对象进行设置证书对象信息。证书对象只有在正确设置之后才能进行其他操作。此函数的内部工作内容：

1. 通过引擎对象传递系统配置文件路径。
2. 通过配置文件初始化根证书对象。
3. 通过配置文件初始化 CRL 对象。

参数说明：

Engine:　　已经初始化过的引擎对象。

返回参数：

成功返回 0，失败返回<0，由 ErrorCode()方法中取错误码，由 GetErrorMessage()方法中取错误信息。

InitCertFromFile　　根据证书文件路径初始化公钥证书

方法声明：

long InitCertFromFile (BSTR sCertFile, long IsBase64Code);

方法说明：

根据指定的证书文件构造并初始化一个证书对象。前提条件：已经成功初始化引擎。默认时，在初始化过程中同时效验证书有效期。

参数说明：

sCertFile:　　指定.cer证书文件完整路径

IsBase64Code:　是否为 BASE64 编码格式

0 = 二进制编码，1 = BASE64 编码

返回参数：

成功返回 0，失败返回其他值。

InitCertFromMem　　根据证书编码数据初始化公钥证书

方法声明：

long InitCertFromMem (BSTR Base64CertData);

方法说明：

根据指定的 BASE64 编码格式的证书数据构造并初始化公钥证书对象。前提条件：已经成功初始化引擎。默认时，在初始化过程中同时效验证书有效期。

参数说明：

Base64CertData:　　BASE64 编码公钥证书数据

返回参数：

成功返回 0，失败返回其他值。

GetPublicCertFromEngine　　从引擎对象初始化公钥证书

方法声明：

long GetPublicCertFromEngine (LPDISPATCH I_Engine, long nCertIndex);

方法说明：

根据指定引擎库中证书次序号从引擎对象中导出并初始化公钥证书对象。

参数说明：

Engine:　　BASE64 编码公钥证书数据

nCertIndex:　　输入引擎库中证书次序号(从 0 开始)

返回参数：

成功返回 0，失败返回其他值。

GetCert　　获得公钥证书 BASE64 编码数据

方法声明：

BSTR GetCert ();

方法说明：

获得公钥证书 BASE64 编码数据。

参数说明：

返回参数：

成功返回公钥证书 BASE64 编码数据，失败返回 NULL。

ViewCertWnd　　显示指定证书窗体

方法声明：

long ViewCertWnd ()

方法说明：

显示证书窗体，前提条件：已经成功初始化证书。

注：此函数在 B/S 结构中作为 Web 服务器脚本中不能使用。

参数说明：

返回参数：

成功返回 0，失败返回其他值。

ViewRootCertWnd　　显示根证书窗体

方法声明：

long ViewRootCertWnd ()

方法说明：

显示根证书窗体，前提条件：已经成功初始化根证书。

注：此函数在 B/S 结构中作为 Web 服务器脚本中不能使用。

参数说明：

返回参数：

成功返回 0，失败返回其他值。

4.5.3 证书解析方法

GetVer　　获得证书版本

方法声明：

long GetVer ();

方法说明：

获得证书的版本号。

参数说明：

返回参数：

成功返回版本号，失败返回<0。

GetVerStr　　获得证书版本信息

方法声明：

BSTR GetVerStr ();

方法说明：

获得证书的版本信息。

参数说明：

返回参数：

成功返回版本信息，失败返回 NULL。

GetCertSN　　获得证书序列号

方法声明：

BSTR GetCertSN ();

方法说明：

获得证书的序列号。

参数说明：

返回参数：

成功返回序列号，失败返回 NULL。

GetSignAlgoOID　　获得证书签名算法 OID

方法声明：

BSTR GetSignAlgoOID ();

方法说明：

获得证书的签名算法 OID。

参数说明：

返回参数：

成功返回签名算法 OID，失败返回 NULL。

GetIssuer　　获得证书颁发者

方法声明：

BSTR GetIssuer ();

方法说明：

获得证书的颁发者。

参数说明：

返回参数：

成功返回颁发者，失败返回 NULL。

GetBeforeValidityStr　　获得证书起始有效期

方法声明：

BSTR GetBeforeValidityStr ();

方法说明：

获得证书的起始有效期。格式：xxxx 年 xx 月 xx 日 hh:mm:ss

参数说明：

返回参数：

成功返回起始有效期，失败返回 NULL。

GetAfterValidityStr　　获得证书终止有效期

方法声明：

BSTR GetAfterValidityStr ();

方法说明：

获得证书的终止有效期。格式：xxxx 年 xx 月 xx 日 hh:mm:ss

参数说明：

返回参数：

成功返回终止有效期，失败返回 NULL。

GetSubject　　获得证书主题

方法声明：

BSTR GetSubject ();

方法说明：

获得证书的主题。

参数说明：

返回参数：

成功返回主题，失败返回 NULL。

GetSubjectHash　　获得证书主题的哈希值

方法声明：

BSTR GetSubjectHash(BSTR sHashAlgo);

方法说明：

获得证书主题的哈希值(BASE64 编码)。

参数说明：

sHashAlgo：指定哈希算法 OID

SHA-1:	"1.3.14.3.2.26"	产生哈希值长度：28字节
MD5:	"1.2.840.113549.2.5"	产生哈希值长度：24字节
MD4:	"1.2.840.113549.2.4"	产生哈希值长度：24字节
MD2:	"1.2.840.113549.2.2"	产生哈希值长度：24字节

返回参数：

成功返回主题的哈希值，失败返回 NULL。

GetPublicKeyAlgoOID　　获得证书主题公钥算法 OID

方法声明：

BSTR GetPublicKeyAlgoOID ();

方法说明：

获得证书的主题公钥算法 OID。如：

"1.2.840.113549.1.1.1" = szOID_RSA_RSA

参数说明：

返回参数：

成功返回主题公钥算法 OID，失败返回 NULL。

GetKeyUsageStr　　获得证书密钥用法信息

方法声明：

BSTR GetKeyUsageStr ();

方法说明：

获得证书的密钥用法信息。如：

签名用法：” Digital Signature(80)”

加密用法：”Key Encipherment, Data Encipherment(30)”

无密钥用法：””

参数说明：

返回参数：

成功返回密钥用法，失败返回 NULL。

GetKeyUsageInt　　获得证书密钥用法值

方法声明：

long GetKeyUsageInt ();

方法说明：

获得证书的密钥用法值。如：

签名用法：128

加密用法：48

无密钥用法：0

参数说明：

返回参数：

成功返回密钥用法值，失败返回<0 值。

GetExtendCount　　获得证书扩展域总数

方法声明：

long GetExtendCount ();

方法说明：

获得证书的扩展域总数。

注意：只有调用此函数后，才能获得扩展域中的 OID 和值。

参数说明：

返回参数：

成功返回扩展域总数，失败返回<0 值。

GetExtendOID　　获得证书扩展域 OID

方法声明：

BSTR GetExtendOID (long nIndex);

方法说明：

获得证书的扩展域 OID。返回的 OID 如：

1.2.86.11.7.1 = 个人身份标识码

1.2.86.11.7.2 = 个人社会保险号

1.2.86.11.7.3 = 企业组织机构代码
1.2.86.11.7.4 = 企业工商注册号
1.2.86.11.7.5 = 企业(国税)号
1.2.86.77.1 = 企业(地税)号

参数说明：

nIndex: 要获得的证书扩展域序号(从 1 开始)

返回参数：

成功返回证书扩展域 OID，失败返回 NULL。

GetExtendItem 获得证书扩展域值

方法声明：

BSTR GetExtendItem (long nIndex);

方法说明：

获得证书的扩展域值。

参数说明：

nIndex: 要获得的证书扩展域序号(从 1 开始)

返回参数：

成功返回证书扩展域值，失败返回 NULL。

GetExtendItem2 获得证书扩展域值

方法声明：

BSTR GetExtendItem2 (BSTR sOID);

方法说明：

获得证书的扩展域值。

参数说明：

sOID: 要获得的证书扩展域 OID

返回参数：

成功返回证书扩展域值，失败返回 NULL。

4.5.4 证书主题_解析方法

GetCertDN_CN 获得证书主题_CN

方法声明：

BSTR GetCertDN_CN ();

方法说明：

获得证书的公共名称 CN。

参数说明：

返回参数：

成功返回证书主题的公共名称 CN，失败返回 NULL。

GetCertDN_OU　　获得证书主题_组织单位

方法声明：

BSTR GetCertDN_OU ();

方法说明：

获得证书主题的组织单位。

参数说明：

返回参数：

成功返回证书组织单位，失败返回 NULL。

GetCertDN_OUa　　获得证书主题_企业代码证号

方法声明：

BSTR GetCertDN_OUa ();

方法说明：

获得证书主题的企业代码证号。

参数说明：

返回参数：

成功返回证书主题的企业代码证号，失败返回 NULL。

GetCertDN_OUi　　获得证书主题_备注项

方法声明：

BSTR GetCertDN_OUi ();

方法说明：

获得证书主题的备注项。

参数说明：

返回参数：

成功返回证书主题的备注项，失败返回 NULL。

GetCertDN_O　　获得证书主题_组织

方法声明：

BSTR GetCertDN_O ();

方法说明：

获得证书主题的组织。

参数说明：

返回参数：

成功返回证书主题的组织，失败返回 NULL。

GetCertDN_L　　获得证书主题_市

方法声明：

BSTR GetCertDN_L ();

方法说明：

获得证书主题的市。

参数说明：

返回参数：

成功返回证书主题的市，失败返回 NULL。

GetCertDN_S　　获得证书主题_省

方法声明：

BSTR GetCertDN_S ();

方法说明：

获得证书主题的省。

参数说明：

返回参数：

成功返回证书主题的省，失败返回 NULL。

GetCertDN_C　　获得证书主题_国家

方法声明：

BSTR GetCertDN_C ();

方法说明：

获得证书主题的国家。

参数说明：

返回参数：

成功返回证书主题的国家，失败返回 NULL。

GetCertDN_E　　获得证书主题_电子邮件

方法声明：

BSTR GetCertDN_E ();

方法说明：

获得证书主题的电子邮件。

参数说明：

返回参数：

成功返回证书主题的电子邮件，失败返回 NULL。

4.5.5 LDAP 操作方法

InitLdap　初始化 LDAP 服务器

方法声明：

long InitLdap (BSTR LdapAddress, long LdapPort)

方法说明：

初始化 LDAP 服务器。只有 LDAP 初始化后，才能从 LDAP 服务器进行其他操作。

参数说明：

szLDAPServerAddress：LDAP 服务器地址 [可选参数]，如果输入””: 默认从 ldap.ln-ca.com 下载

dwPort：LDAP 服务器端口 [可选参数]，如果输入<=0: 默认从 391 下载

返回参数：

成功返回 0，失败返回其他值。

GetCertStatusFromLdap　从 Ldap 服务器获得证书状态

方法声明：

long GetCertStatusFromLdap (BSTR CertSN)

方法说明：

从 Ldap 服务器获得证书状态。

前提条件：成功地从 LDAP 服务器查询过证书，并且证书已经被初始化。

证书状态。

3：新申请（证书未下载）

5：证书使用中

6：证书注销

参数说明：

CertSN：　要查询的证书序列号

返回参数：

成功返回证书状态号，失败返回<0。

GetCertTypeFromLdap　从 Ldap 服务器获得证书类型

方法声明：

long GetCertTypeFromLdap (BSTR CertSN)

方法说明：

从 Ldap 服务器获得证书类型。

前提条件：成功地从 LDAP 服务器查询过证书，并且证书已经被初始化。

目前设定的证书种类如下：

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

参数说明：

CertSN：　要查询的证书序列号

返回参数：

成功返回证书类型号，失败返回<0。

ReadCertFromLdap　从 Ldap 服务器获得公钥证书

方法声明：

```
BSTR ReadCertFromLdap (BSTR CertSN,
                       BSTR CertFile,
                       long IsSaveBase64Code);
```

方法说明：

从 Ldap 服务器获得公钥证书 BASE64 编码数据，同时将公钥证书保存到本地文件。

参数说明：

CertSN：　要查询的证书序列号
CertFile：　要保存的公钥证书文件名(完整路径)
IsSaveBase64Code：是否保存为 BASE64 编码格式
　0 = 二进制编码，1 = 　BASE64 编码

返回参数：

成功返回公钥证书 BASE64 编码数据，失败返回 NULL，由 GetErrorCode 方法中取错误码，由 GetErrorMessage 方法中取错误信息。

ReadRootCertFromLdap　从 Ldap 服务器获得公钥根证书

方法声明：

BSTR ReadRootCertFromLdap (BSTR RootFile, long IsSaveBase64Code);

方法说明：

从 Ldap 服务器获得公钥根证书 BASE64 编码数据，同时将公钥根证书保存到本地文件。

参数说明：

RootFile：　要保存的公钥根证书文件名(完整路径)

IsSaveBase64Code：是否保存为 BASE64 编码格式

0 = 二进制编码，1 = BASE64 编码

返回参数：

成功返回公钥根证书 BASE64 编码数据，失败返回 NULL，由 GetErrorCode 方法中取错误码，由 GetErrorMessage 方法中取错误信息。

ReadCRLFromLdap　从 Ldap 服务器获得 CRL 吊销列表

方法声明：

BSTR ReadCRLFromLdap (BSTR CRLFile, long IsSaveBase64Code);

方法说明：

从 Ldap 服务器获得 CRL 吊销列表 BASE64 编码数据，同时将 CRL 吊销列表保存到本地文件。

参数说明：

CRLFile：　要保存的 CRL 吊销列表文件名(完整路径)

IsSaveBase64Code：是否保存为 BASE64 编码格式

0 = 二进制编码，1 = BASE64 编码

返回参数：

成功返回 CRL 吊销列表 BASE64 编码数据，失败返回 NULL，由 GetErrorCode 方法中取错误码，由 GetErrorMessage 方法中取错误信息。

4.5.6 证书效验操作方法

InitRootCert　初始化根证书

方法声明：

long InitRootCert (BSTR sRootCertFile);

方法说明：

通过本地二进制编码根证书文件初始化根证书对象。

参数说明：

sRootCertFile：指定二进制编码根证书文件(全路径)

返回参数：

成功返回 0，失败返回其他值。

InitCRL　　初始化 CRL 吊销列表

方法声明：

long InitCRL (BSTR sCRL, long nMode, long IsSaveBase64);

方法说明：

通过本地二进制编码 CRL 吊销列表文件初始化 CRL 吊销列表对象。分两种方式初始化：

在线方式：直接从 Ldap 服务器下载，初始化 CRL 对象，并保存到 sCRL 指定的文件中。

离线方式：从 sCRL 文件中读二进制编码数据，初始化 CRL 对象。

参数说明：

sCRL： 指定二进制编码 CRL 吊销列表文件(全路径)

nMode：CRL 吊销列表方式

0 = 在线，1 = 离线

IsSaveBase64：当 nMode=1 为在线方式时，保存 CRL 编码格式，可选参数：默认 0

0 = 二进制编码， 1 = Base64 编码

返回参数：

成功返回 0，失败返回其他值。

RootCertVerifyCRL　　根证书验证 CRL

方法声明：

long RootCertVerifyCRL (long IsVerifyTime, BSTR sSystemTime);

方法说明：

使用根证书验证 CRL。前提条件：成功初始化 CRL.

参数说明：

IsVerifyTime：是否效验 CRL 时间，(当=1 时，sSystemTime 参数有效)

0 = 不校验 CRL 时间

1 = 用本机系统时间校验 CRL 时间

2 = 校验 CRL 时间(下面的参数有效)

sSystemTime：指定日期和时间 [可选参数]，默认:：””.

格式：yyyy-mm-dd hh:mm:ss

返回参数：

0=有效，其它=无效

CRLVerifyCert　　CRL 验证证书

方法声明：

long CRLVerifyCert ();

方法说明：

使用 CRL 验证证书。前提条件：成功初始化 CRL、被验证证书。

参数说明：

返回参数：

0=未吊销，1=吊销

RootCertVerifyCert　　根证书验证证书

方法声明：

long RootCertVerifyCert (long lVerifyFlag);

方法说明：

使用根证书验证证书。前提条件：成功初始化根证书、被验证证书。

参数说明：

lVerifyFlag：　　验证方式

0: 校验 LNCA 根证书和有效期

1: 只校验 LNCA 根证书

2: 只校验有效期

其他: 校验 LNCA 根证书和有效期

返回参数：

0=有效，其它=无效

4.5.7 其他操作方法

GetErrorCode　　获取错误代码

函数声明：

long ErrorCode();

方法说明：

获取错误号，在程序出错的时候，调用本方法。

参数说明：

返回值：

0 表示成功；<0 值为最后发生的错误号码。

GetErrorMessage　　获取错误信息

函数声明：

BSTR GetErrorMessage ();

方法说明：

获取错误信息，在程序出错的时候，调用本方法。

参数说明：

返回值：

4.5.8 ASP 中调用证书接口

```
Dim objCertificate
Set objCertificate = Server.CreateObject("LNCACryptoCom.LNCACertificateCom.1")
Ret = objCertificate. SetEngineInfo(objEngine)
nErrorCode = objCertificate.ErrorCode
If 0 <> nErrorCode Then
    Response.Write "初始化证书对象失败！ " & "ErrorCode = " & nErrorCode & "<br>"
    Response.End()
Else
    Response.Write("初始化证书对象成功.<br>")
    Set Session("obj_Certificate") = objCertificate
End If
```

5 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_LIBARY   "找不到动态库"
#define E_CAN_NOT_FIND_LIBARY       -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

#define E_MSG_ENGINE_INVALIDATE         "引擎未初始化"
#define E_ENGINE_INVALIDATE              -6027

#define E_MSG_VERIFY_PASSWORD_FAILED "校验口令失败"
#define E_VERIFY_PASSWORD_FAILED        -6028

#define E_MSG_VERIFY_CRL_FAILED         "校验 CRL 失败"
#define E_VERIFY_CRL_FAILED             -6029

#define E_MSG_VERIFY_ROOT_FAILED        "校验根证书失败"
#define E_VERIFY_ROOT_FAILED            -6030

#define E_MSG_FILE_NOT_EXIST_FAILED     "文件不存在"
#define E_FILE_NOT_EXIST_FAILED         -6031

#define E_MSG_FILE_NOT_WRITE_FAILED     "不允许写文件"
#define E_FILE_NOT_WRITE_FAILED         -6032
```

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。

6 历史版本

2006.06.20　v1.6.5.0

1. 增加输入和输出数据格式控制
2. 显示错误模式等属性，去掉了部分函数中关于输入和输出数据格式控制参数
3. 修改了证书管理结构。
4. 增加通过哈希数据产生数字签名函数

2006.05.15　v1.6.0.0　增加对文件的哈希和签名哈希操作，修改初始化 CRL 函数中的在线功

能，增加一个可选参数。

2006.04.15 v1.5.0.0 对部分签名、RSA 加密、对称加密和产生随机数函数的参数进行改进，增加了可选参数。

2006.04.01 v1.3.0.0 增加证书主题的哈希值、BASE64 编解码文件操作方法。

2006.03.16 v1.2.0.0 修改证书效验函数中的 Bug。

2006.03.15 v1.1.0.0 修改引擎接口中有关证书数组管理方式和.PFX 文件证书使用等。

2006.01.20 v1.0.0.1 Com 的第一个版本。