Руководство разработчика приложений C++
Функции библиотеки CesarisApi.dll

Функция `GetListCert` открывает диалоговое окно для выбора сертификата из списка

Синтаксис:

```
HRESULT GetListCert(IN char* pchStoreName,
                    IN char* pchCaption,
                    IN char* pchText,
                    IN DWORD dwDontUseColumn,
                    OUT BYTE** ppbCertificateOut,
                    OUT DWORD* pdwCertSizeOut);
```

Параметры:

pchStoreName – имя хранилища из которого будут выбираться сертификаты (CA,MY,ROOT,SPC)..

pchCaption - заголовок диалогового окна

pchText - дополнительный текст внутри окна

dwDontUseColumn – флаги которые могут комбинироваться для отображения необходимых колонок (определены в `cryptuiapi.h`)

Value	Meaning
CRYPTUI_SELECT_ISSUEDTO_COLUMN	Do not display the ISSUEDTO information.
CRYPTUI_SELECT_ISSUEDBY_COLUMN	Do not display the ISSUEDBY information.
CRYPTUI_SELECT_INTENDEDUSE_COLUMN	Do not display IntendedUse information.
CRYPTUI_SELECT_FRIENDLYNAME_COLUMN	Do not display the display name information.
CRYPTUI_SELECT_LOCATION_COLUMN	Do not display location information.
CRYPTUI_SELECT_EXPIRATION_COLUMN	Do not display expiration information.

ppbCertificateOut – указатель на массив байт который представляет выбранный сертификат

pdwCertSizeOut - длина возвращаемого массива.

Функция `GetCertByThumbprint` возвращает сертификат по его отпечатку

Синтаксис:

```
HRESULT GetCertByThumbprint(IN char* pchStoreName,
                            IN char* chThumbprint,
                            OUT BYTE** ppbCertificateOut,
                            OUT DWORD* pdwCertSizaOut);
```

Параметры:

pchStoreName - имя хранилища, из которого будут выбираться сертификаты (CA,MY,ROOT,SPC)..

chThumbprint – отпечаток сертификата по которому будет осуществляться поиск.
ppbCertificateOut – указатель на массив байт, который представляет выбранный сертификат.
pdwCertSizaOut - длина возвращаемого массива

Предупреждение: отпечаток сертификата – это хеш- значение сертификата, включая его «Понятное /дружественное имя» (FRIENDLYNAME). При изменении этого имени, изменяется и отпечаток.

Функция `Hash` вычисляет хеш значение, используя хеш-алгоритм, указанный в алгоритме подписи сертификата. Например,

sha1RSA - хеш-алгоритм есть SHA1
DSTU 4145 - хеш-алгоритм есть ГОСТ 34311

Синтакис:

```
HRESULT Hash(IN BYTE* pbCertificate,
             IN DWORD dwCertSize,
             IN BYTE* pbData,
             IN DWORD dwDataSize,
             OUT BYTE** ppbHashedData,
             OUT DWORD* pdwHashedDataSize);
```

Параметры:

pbCertificate – сертификат, на хеш- алгоритме которого будет вычислено хеш- значение. Этот сертификат можно получить с помощью функций `GetCertByThumbprint или GetListCert.`
dwCertSize – длина массива байт представляющего сертификат.
pbData – блок данных, для которого необходимо вычислить хеш
dwDataSize – длина блока данных.
ppbHashedData – указатель на результат вычисления хеш
pdwHashedDataSize – длина хэша

Функция `SignPKCS7` создает цифровую подпись в формате PKCS#7, не содержащую подписываемые данные `(режим detached)`. Результат вычисления - ASN.1 структура в формате PKCS#7 (signedData, OID=1.2.840.113549.1.7.2), содержащая сертификат подписи.

Синтаксис:

```
HRESULT SignPKCS7(IN BYTE* pbCertificate,
                  IN DWORD dwCertSize,
                  IN BYTE* pbData,
                  IN DWORD dwDataSize,
                  IN char* pchPassword,
                  IN bool fDetached,
                  OUT BYTE** ppbSignature,
                  OUT DWORD* pdwSignSize);
```

Праметры:

pbCertificate – сертификат подписи.
dwCertSize – длина сертификата подписи.
pbData – данные, которые необходимо подписать.
dwDataSize – длина данных, которые необходимо подписать.

pchPassword – пароль к ключу подписи. Может быть установлен в NULL, тогда будет всплывать диалоговое окно для ввода пароля, если такой был установлен.
fDetached – показатель включаемости данных в сообщение (true – detached, false – attached)
ppbSignature – вычисленная цифровая подпись в формате PKCS#7.
pdwSignSize – длина подписи.

Функция `VerifyPKCS7` проверка подписи. Подпись должна иметь формат PKCS#7. Возможны режимы `detached и attached`.

Синтаксис:

```
HRESULT VerifyPKCS7(IN BYTE* pbData,
                    IN DWORD dwDataSize,
                    IN BYTE* pbSignature,
                    IN DWORD dwSignatureSize);
```

Праметры:

pbData – данные, которые были подписаны (в случае `attached` - NULL).
dwDataSize – размер данных (в случае `attached` - 0).
pbSignature – цифровая подпись
dwSignatureSize – длина подписи.

Результат: Если проверка подписи прошла удачно - возвращен 0.

Функция `Sign` подписывает блок данных на ключе заданного сертификата. Результат вычесления - масив байт, подпись в формате алгоритма подписи (для ГОСТ 34.310 – это ASN1 BitString).

Синтаксис:

```
HRESULT Sign(IN BYTE* pbCertificate,
             IN DWORD dwCertSize,
             IN BYTE* pbData,
             IN DWORD dwDataSize,
             IN char* pchPassword,
             OUT BYTE** ppbSignedData,
             OUT DWORD* pdeSignedDataSiza);
```

Параметры:
pbCertificate – сертификат, на ключе которого буду подписаны данные.
dwCertSize – длина массива байт представляющего сертификат.
pbData – блок данных, который необходимо подписать.
dwDataSize- длина массива данных.
pchPassword – пароль (PIN) доступа к приватному ключу. Параметр может быть установлен в NULL, тогда если требуется пароль (PIN), то провайдером будет поднято диалоговое окно для ввода пароля (PIN).
ppbSignedData – массив данных, который является цифровой подписью (для ГОСТ 34.310 – это ASN1 BitString).
pdeSignedDataSiza – длина подписи.

Примечание:

1. Подпись возвращается в формате алгоритма подписи. Так для алгоритмов RSA и ДСТУ – это массив байт (непосредственно значение подписи). Для ГОСТ 34.310 возвращается значение VALUE в виде ASN1 BitString с лидирующими нулем(и) (unusedBits, один или два нуля), т.е. в формате как для подписи сертификата; внутри VALUE элемент SEQUENCE.

   Но, например, для SMIME структура подписи ГОСТ другая - не требуется ASN1 BitString с лидирующими нулем(и), а требуется непосредственно SEQUENCE. Поэтому в таких случаях надо отбросить лидирующие нули в значении подписи.
2. Пароль (PIN) при использовании провайдеров Microsoft, если не включена опция «Усиленная защита», не требуется, т.к. пользователь уже аутентифицирован при входе в систему и доступ к его приватному ключу выполняется без ввода пароля.

Функция `VerifySign` выполняет проверку цифровой подписи, созданной при помощи функции Sign.

Синтаксис:

```
HRESULT VerifySign(IN BYTE* pbCertificate,
                           IN DWORD dwCertSize,
                           IN BYTE* pbData,
                           IN DWORD dwDataSize,
                           IN BYTE* pbSignature,
                           IN DWORD dwSignatureSize);
```

Праметры:

dwCertSize – сертификат, на ключе которого была создана цифровая подпись.
dwCertSize – длина масива байт, представляющего сертификат.
pbData – данные, для которых создавалась подпись.
dwDataSize – длина блока данных.
pbSignature – масив байт, представляющий цифровую подпись вычисленную при помощи функции Sign.
dwSignatureSize – длина подписи.

Результат: Проверка подписи прошла удачно если возвращен 0.

Функция `GetCspProviders` возвращает список установленных криптопровайдеров в виде строки, элементы которой разделены символом ”|”.

Синтаксис:

```
HRESULT GetCspProviders(OUT char** ppCspList);
```

Параметры:

ppCspList – указатель на строку со списком провайдеров

Функция `GetSCCspProviders` возвращает список установленных криптопровайдеров смарт-карт, в виде строки, елементы которой разделены символом ”|”.

Синтаксис:

```
HRESULT GetSCCspProviders(OUT char** ppCspList);
```

Параметры:

ppCspList – указатель на строку со списком провайдеров.

Функция `ImportPFX` импортирует объекты (приватный и открытый ключ, сертификат) из файла PFX (PKCS#12) в хранилище криптопровайдера MS Provider (только для RSA). Предназначена для установки RSA сертификата сервера приложений, который работат от имени SYSTEM.

Синтаксис:

```
HRESULT ImportPFX(IN char* pchStoreName,
                                IN BYTE* pbPfxData,
                                IN DWORD dwPfxDataSiza,
                                IN char* pchPassword,
                                IN DWORD dwFlags);
```

Параметры:

pchStoreName – имя хранилища в которое необходимо импортировать сертификат (CA,MY,ROOT,SPC)..

pbPfxData – данные из файла PFX.

dwPfxDataSiza – размер данных.

pchPassword – пароль к файлу PFX, если был установлен.

dwFlags – кобинация флагов, которые определены в файле WinCrypt.h, будут применены к личному ключу.Могут применятся комбинации из следующих флагов сложеных по или: CRYPT_EXPORTABLE, CRYPT_USER_PROTECTED, CRYPT_MACHINE_KEYSET , CRYPT_USER_KEYSET.

Функция `CreateRequest` генерирует ключевую пару и запрос PKCS#10 на получение сертификата. Чтобы использовать эту функцию, должен быть установлен COM обьект CEnroll `(xenroll.dll,` включен в инсталляцию Windiws XP/ 2000 и выше`)`.

Синтаксис:

```
HRESULT CreateRequest(IN char* pchCspName,
              IN DWORD dwKeySize,
              IN char* pchDnName,
              IN char* pchFriendlyName,
              OUT char** ppchRequest);
```

Параметры:

pchCspName – имя криптопровайдера для генерации ключевой пары; это имя можно получить с помощью функций GetCspProviders или GetSCCspProviders.

dwKeySize – длина ключа, который будет сгенерирован (в битах).

pchDnName – DN имя сертификата (пример ”CN=MyCertificate O=MyOrg”).

pchFriendlyName – дружественное имя, которое может быть добавлено к сертификату. Смарт- карты этот параметр могут не поддерживать. Параметр не обязателен.

ppchRequest – возвращает запрос на сертификат в виде строки Base64.

Функция `InstallCertificate` устанавливает полученный сертификат от центра сертификации, запрос на который был получен с помощью функции `CreateRequest`.

Синтаксис:

```
HRESULT InstallCertificate(IN char* pchCertificate);
```

Параметры:

pchCertificate – сертификат X509 в виде строки base64.

Функция `EncryptPkcs7` шифрует данные. Результат выполнения – ASN.1 структура в формате PKCS#7 (smime-type=enveloped-data; OID= 1.2.840.113549.1.7.3) с включенным в структуру сертификатом шифрования.

Синтаксис:

```
HRESULT EncryptPkcs7(IN BYTE* pbCertificate,
                     IN DWORD dwCertSize,
                     IN BYTE* pbData,
                     IN DWORD dwDataSize,
                     OUT BYTE** ppbEncryptedData,
                     OUT DWORD* pdwEncryptedDataSize);
```

Параметры:

pbCertificate – сертификат шифрования, на публичном ключе которого будут шифроваться данные.
dwCertSize – длина сертификата.
pbData – данные, которые шифруются.
dwDataSize – длина данных.
ppbEncryptedData – указатель на масив байт с зашифроваными данными.
pdwEncryptedDataSize – длина зашифрованных данных.

Функция `DecryptPkcs7` расшифровывает зашифрованные данные. Данные должны иметь структуру PKCS#7 (smime-type=enveloped-data; OID= 1.2.840.113549.1.7.3).

Синтаксис:

```
HRESULT DecryptPkcs7(IN BYTE* pbCertificate,
                     IN DWORD dwCertSize,
                     IN BYTE* pbEncryptedData,
                     IN DWORD dwEncryptedDataSize,
                     IN char* pchPassword,
                     OUT BYTE** ppbDecryptData,
                     OUT DWORD* pdwDecryptDataSize);
```

Параметры:

pbCertificate – сертификат, на открытом ключе которого были зашифрованны данные. Этот параметр необходимо указывать (в паре с параметром pchPassword) только в том случае, если нужно избежать появления диалогового окна для ввода пароля к хранилищу, в противном случае установить NULL. Если же какой-то из параметров pbCertificate или pchPassword не будет указан, то будет поднято диалоговое окно.
dwCertSize – длина сертификата.
pbEncryptedData – зашифрованные данные.
dwEncryptedDataSize – длина зашифрованных данных.
pchPassword – пароль к приватному ключу сертификата шифрования. Этот параметр необходимо указывать только в том случае, если нужно избежать появления диалогового окна для ввода пароля к приватному ключу сертификата шифрования, в противном случае установить NULL. Параметр задается в паре с pbCertificate.
ppbDecryptData – указатель на масив расшифрованных данных.
pdwDecryptDataSize – длина расшифрованных данных.

Функция: GetCertListByFilter выводит стандартное диалоговое окно выбора сертификатов по фильтру,

Синтаксис:

HRESULT GetCertListByFilter(IN char* pchStoreName,
IN char* pchCaption,
IN char* pchText,
IN DWORD dwDontUseColumn,
OUT BYTE** ppbCertificateOut,
OUT DWORD* pdwCertSizaOut);

Параметры:

pchStoreName – имя хранилища из которого будут выбираться сертификаты (CA,MY,ROOT,SPC)..

pchCaption - заголовок диалогового окна

pchText - дополнительный текст внутри окна

dwDontUseColumn – флаги которые могут комбинироваться для отображения необходимых колонок (см. функцию `GetListCert`)

pchFilterEmail – e-mail по которому будут отбираться сертификаты

dwFilterCertType - дополнительные параметры для отбора сертификатов (определены в CesarisApi.h)

`FILTER_CERT_ALL`	0
`FILTER_CERT_SIGN`	1
`FILTER_CERT_ENCRYPT`	2
`FILTER_CERT_RSA`	4
`FILTER_CERT_GOST`	8
`FILTER_CERT_DSTU`	16

`ppbCertificateOut` - указатель на массив байт который представляет выбранный сертификат

pdwCertSizaOut - длина возвращаемого массива

Функция `EncryptDH` шифрует данные. Результат выполнения – ASN.1 структура в формате DER (smime-type=enveloped-data; OID= 1.2.840.113549.1.7.3).

Синтаксис:

```
HRESULT EncryptDH(IN const BYTE* pbIssCertificate,
                  IN DWORD dwIssCertSize,
                  IN const BYTE* pbRecCertificate,
                  IN DWORD dwRecCertSize,
                  IN const BYTE* pbData,
                  IN DWORD dwDataSize,
                  IN const char* pchPassword,
                  OUT BYTE** ppbEncryptedData,
                  OUT DWORD* pdwEncryptedDataSize);
```

Параметры:

pbIssCertificate – сертификат отправителя.

dwIssCertSize – длина сертификата отправителя.

pbRecCertificate – сертификат получателя.
dwRecCertSize – длина сертификата получателя.
pbData – данные, которые шифруются.
dwDataSize – длина данных.
pchPassword - пароль к приватному ключу сертификата отправителя. Этот параметр необходимо указывать только в том случае, если нужно избежать появления диалогового окна для ввода пароля к приватному ключу сертификата отправителя, в противном случае установить NULL.
ppbEncryptedData – указатель на масив байт с зашифрованными данными.
pdwEncryptedDataSize – длина зашифрованных данных.

Функция `DecryptDH` расшифровывает зашифрованные данные. Данные должны иметь структуру DER (smime-type=enveloped-data; OID= 1.2.840.113549.1.7.3).

Синтаксис:

```
HRESULT DecryptDH(IN const BYTE* pbIssCertificate,
                  IN DWORD dwIssCertSize,
                  IN const BYTE* pbRecCertificate,
                  IN DWORD dwRecCertSize,
                  IN const BYTE* pbEncryptedData,
                  IN DWORD dwEncryptedDataSize,
                  IN const char* pchPassword,
                  OUT BYTE** ppbDecryptData,
                  OUT DWORD* pdwDecryptDataSize);
```

Параметры:
pbIssCertificate – сертификат отправителя.
dwIssCertSize – длина сертификата отправителя.
pbRecCertificate – сертификат получателя.
dwRecCertSize – длина сертификата получателя.
pbEncryptedData – зашифрованные данные.
dwEncryptedDataSize – длина зашифрованных данных.
pchPassword – пароль к приватному ключу сертификата получателя. Этот параметр необходимо указывать только в том случае, если нужно избежать появления диалогового окна для ввода пароля к приватному ключу сертификата получателя, в противном случае установить NULL.
ppbDecryptData – указатель на масив расшифрованных данных.
pdwDecryptDataSize – длина расшифрованных данных.

Функция `StreamInitializeEncryptDH` генерирует ключ для шифрования данных, сохраняет его в памяти (для дальнейшего использования функцией `StreamUpdateEncryptDH`) и зашифровывает. На выход отдает зашифрованный ключ шифрования данных и публичный ключ отправителя (в случае, если при шифровании ключа шифрования данных использовался динамический механизм).

Синтаксис:

```
HRESULT StreamInitializeEncryptDH( IN const BYTE* pbIssCertificate,
                                   IN DWORD dwIssCertSize,
                                   IN const BYTE* pbRecCertificate,
                                   IN DWORD dwRecCertSize,
```

```
                                IN const char* pchPassword,
                                OUT BYTE** ppbPublicKey,
                                OUT DWORD* pdwPublicKeySize,
                                OUT BYTE** ppbEncryptedKey,
                                OUT DWORD* pdwEncryptedKeySize);
```

Параметры:
pbIssCertificate – сертификат отправителя.
dwIssCertSize – длина сертификата отправителя.
pbRecCertificate – сертификат получателя.
dwRecCertSize – длина сертификата получателя.
pchPassword - пароль к приватному ключу сертификата отправителя. Этот параметр необходимо указывать только в том случае, если нужно избежать появления диалогового окна для ввода пароля к приватному ключу сертификата отправителя, в противном случае установить NULL.
ppbPublicKey – публичный ключ отправителя. Этот параметр заполняется функцией только если алгоритм ключевой пары сертификата отправителя отличается от алгоритма ключевой пары сертификата получателя – в таком случае функция генерирует новую ключевую пару для отправителя.
pdwPublicKeySize – длина публичного ключа.
ppbEncryptedKey – зашифрованный ключ шифрования данных.
pdwEncryptedKeySize – длина зашифрованного ключа.

Функция `StreamUpdateEncryptDH` зашифровывает блок данных на ключе, сгенерированным функцией `StreamInitializeEncryptDH`.

Синтаксис:
```
HRESULT StreamUpdateEncryptDH(IN const BYTE* pbData,
                              IN DWORD dwDataSize,
                              OUT BYTE** pbEncryptedData,
                              OUT DWORD* dwEncryptedDataSize);
```

Параметры:
pbData – данные, которые шифруются.
dwDataSize – длина данных.
ppbEncryptedData – указатель на масив байт с зашифрованными данными.
pdwEncryptedDataSize – длина зашифрованных данных.

Функция `StreamInitializeDecryptDH` расшифровывает ключ для шифрования данных и сохраняет его в памяти (для дальнейшего использования функцией `StreamUpdateDecryptDH`).

Синтаксис:
```
HRESULT StreamInitializeDecryptDH( IN const BYTE* pbIssCertificate,
                                   IN DWORD dwIssCertSize,
                                   IN const BYTE* pbRecCertificate,
                                   IN DWORD dwRecCertSize,
                                   IN const char* pchPassword,
                                   IN const BYTE* ppbPublicKey,
                                   IN DWORD pdwPublicKeySize,
```

```
                                    IN const BYTE* ppbEncryptedKey,
                                    IN DWORD pdwEncryptedKeySize);
```

Параметры:
pbIssCertificate – сертификат отправителя.
dwIssCertSize – длина сертификата отправителя.
pbRecCertificate – сертификат получателя.
dwRecCertSize – длина сертификата получателя.
pchPassword - пароль к приватному ключу сертификата получателя. Этот параметр необходимо указывать только в том случае, если нужно избежать появления диалогового окна для ввода пароля к приватному ключу сертификата получателя, в противном случае установить NULL.
ppbPublicKey – публичный ключ отправителя. Этот параметр указывается только в случае если при шифровании ключа шифрования данных использовался динамический механизм, т.е. если функция `StreamInitializeEncryptDH` вернула ненулевое значение публичного ключа.
pdwPublicKeySize – длина публичного ключа.
ppbEncryptedKey – зашифрованный ключ шифрования данных.
pdwEncryptedKeySize – длина зашифрованного ключа.

Функция `StreamUpdateDecryptDH` расшифровывает блок данных на ключе, полученным функцией `StreamInitializeDecryptDH`.

Синтаксис:
```
HRESULT StreamUpdateDecryptDH(IN const BYTE* pbEncryptedData,
                              IN DWORD dwEncryptedDataSize,
                              OUT BYTE** ppbDecryptData,
                              OUT DWORD* pdwDecryptDataSize);
```

Параметры:
ppbEncryptedData – зашифрованые данными.
pdwEncryptedDataSize – длина зашифрованных данных.
ppbDecryptData – указатель на масив байт с расшифроваными данными.
pdwDecryptDataSize – длина расшифрованных данных.

Функция `StreamFinalizeDH` удаляет из памяти ключи, занесенные туда функциями `StreamInitializeEncryptDH` и `StreamInitializeDecryptDH`. Должна вызываться после окончания шифрования и расшифрования блоков данных функциями `StreamUpdateEncryptDH` и `StreamUpdateDecryptDH`.

Синтаксис:
```
HRESULT StreamFinalizeDH();
```

Функция `GetOCSPRequest` формирует OCSP запрос.

Синтаксис:
```
HRESULT GetOCSPRequest(  IN BYTE* pbCert,
```

```
                        IN long bCertLen,
                        OUT BYTE* pbOcspRequest,
                        IN,OUT long* bOcspRequestLen,
                        IN DWORD dwFlag)
```

Параметры:

`pbCert` – сертификат.

`bCertLen` – длина зашифрованных данных.

`pbOcspRequest` – указатель на масив байт с OCSP запросом (возможно значение NULL, при этом, в параметре `bOcspRequestLen` будет возвращен необходимый размер OCSP запроса, код возврата при этом `CESARIS_E_BUFFER_TO_SMALL`).

`bOcspRequestLen` – размер OCSP запроса.

`dwFlag` – поле флагов, может быть комбинацией следующих:

Значение	Описание
GET_REQUEST_WITHOUT_SIGN 0	Формирование неподписанного запроса
GET_REQUEST_WITH_NULL_PARAM_IN_ALGORITHM_ID 4	Формирование запроса с параметром алгоритма установленным в NULL
GET_REQUEST_WITH_HASH_GOST 8	В качестве хэш-алгоритма - ГОСТ34311
GET_REQUEST_WITH_D4145PB 16	Устанавливает алгоритм подписи OCSP ответа в ДСТУ-4145 ПБ с хешем ГОСТ-34311
GET_REQUEST_WITH_D4145ONB 32	Устанавливает алгоритм подписи OCSP ответа в ДСТУ-4145 ОНБ с хешем ГОСТ-34311
GET_REQUEST_WITH_RSA_with_SHA1 128	Устанавливает алгоритм подписи OCSP ответа в RSA с хешем SHA1

Возвращаемое значение:

Код возврата	Описание
CESARIS_E_SUCCESS 0	Успех
CESARIS_E_PARAMETER_INVALID -1003	Неверные аргументы
CESARIS_E_BUFFER_TO_SMALL -1002	Недостаточно места для OCSP запроса

Функция `ParseOCSPResponse` возвращает различные элементы OCSP ответа.

Синтаксис:

```
HRESULT ParseOCSPResponse(IN BYTE* pbOCSPResponse,
                          IN LONG bOCSPResponseLen,
                          OUT BYTE* pDerAnswer,
                          IN,OUT LONG* pDerAnswerLen,
                          IN LONG dwFlag);
```

Параметры:

`pbOCSPResponse` – OCSP ответ.

`bOCSPResponseLen` – длина OCSP ответа.

`pDerAnswer` – указатель на масив байт куда будет скопировано значение искомого элемента OCSP ответа (возможно значение NULL, при этом, в параметре `pDerAnswerLen` будет возвращен необходимый размер).

pDerAnswerLen – размер элемента.

dwFlag – поле флагов, может быть одним из следующих:

Значение	Описание
Response_Get_responseStatus 0	Получение статуса OCSP ответа
Response_Get_certStatus 1	Получение статуса сертификата
Response_Get_producedAt 2	Получение времени подписания OCSP ответа
Response_Get_thisUpdate 3	Получение времени установления статуса сертификата (в случае использования онлайн-режима совпадает с Response_Get_producedAt)
Response_Get_nextUpdate 4	Получение времени, когда будет доступна новая информация о статусе сертификата (в случае использования CRL)
Response_Get_revocationTime 5	Получение времени отзыва сертификата
Response_Get_signature 6	Получение подписи OCSP ответа
Response_Get_certificateOCSP 7	Получение сертификата подписи
Response_Get_certSerialNumber 8	Получение серийного номера сертификата подписи
Response_Get_ocspNonce 9	Получение случайной последовательности (nonce)

Возвращаемое значение:

Код возврата	Описание
ASN1_SUCCESS 0	Успех
ASN1_FILE_NOT_FOUND 1	Файл не найден
ASN1_ELEMENT_NOT_FOUND 2	Элемент не найден
ASN1_IDENTIFIER_NOT_FOUND 3	Идентификатор не найден
ASN1_DER_ERROR 4	Внутренняя ошибка
ASN1_VALUE_NOT_FOUND 5	Значение не найдено
ASN1_GENERIC_ERROR 6	Ошибка создания элемента
ASN1_VALUE_NOT_VALID 7	Неверное значение
ASN1_TAG_ERROR 8	
ASN1_TAG_IMPLICIT 9	
ASN1_ERROR_TYPE_ANY 10	
ASN1_SYNTAX_ERROR 11	Синтаксическая ошибка
ASN1_MEM_ERROR 12	Недостаточно памяти
ASN1_MEM_ALLOC_ERROR 13	Ошибка выделения памяти
ASN1_DER_OVERFLOW 14	
ASN1_NAME_TOO_LONG 15	

ASN1_ARRAY_ERROR 16	
ASN1_ELEMENT_NOT_EMPTY 17	

Функция VerifyOCSPResponse проверяет подпись OCSP ответа.

Синтаксис:

```
HRESULT VerifyOCSPResponse(   IN BYTE* pbOCSPResponse,
                              IN LONG pOCSPResponseLen)
```

Параметры:

pbOCSPResponse – OCSP ответ.

bOCSPResponseLen – длина OCSP ответа.

Возвращаемое значение:

В случае успешной проверки возвращает CESARIS_E_SUCCESS (0).

Функция ParseTSPResponse возвращает различные элементы TSP ответа.

Синтаксис:

```
HRESULT ParseTSPResponse(IN BYTE* pbTSPResponse,
                         IN LONG bOTPResponseLen,
                         OUT BYTE* pDerAnswer,
                         IN,OUT LONG* pDerAnswerLen,
                         IN LONG dwFlag);
```

Параметры:

pbTSPResponse – TSP ответ.

bTSPResponseLen – длина TSP ответа.

pDerAnswer – указатель на масив байт куда будет скопировано значение искомого элемента TSP ответа (возможно значение NULL, при этом, в параметре pDerAnswerLen будет возвращен необходимый размер).

pDerAnswerLen – размер элемента.

dwFlag – поле флагов, может быть одним из следующих:

Значение	Описание
TSPResponse_status 0	Получение статуса TSP ответа
TSPResponse_failInfo 1	Получение причины отказа
TSPResponse_policy 2	Получение политики TSP ответа
TSPResponse_genTime 3	Получение времени генерации TSP ответа
TSPResponse_nonce 4	Получение случайной последовательности (nonce)

Возвращаемое значение: Аналогично ParseOCSPResponse.

Функция `VerifyTSPResponse` проверяет подпись TSP ответа.

Синтаксис:

```
HRESULT VerifyTSPResponse(    IN BYTE* pbTSPResponse,
                              IN LONG pTSPResponseLen)
```

Параметры:

`pbTSPResponse` – TSP ответ.

`bTSPResponseLen` – длина TSP ответа.

Возвращаемое значение:

В случае успешной проверки возвращает `CESARIS_E_SUCCESS` (0).

Функции могут возвращать стандартные ошибки Windows а также дополнительные ошибки, основные коды которых приведены ниже:

Код ошибки	Описание
0	Ошибок нет
Общие	
1001	Ошибка памяти
1002	Буфер мал
1003	Ошибочные параметры
1004	Неверный пароль
1005	Ошибочный OID
1049	Неизвестная ошибка
file (1050-1069)	
1051	Файл не найден
1052	Ошибка открытия файла
1053	Ошибка размера файла
1054	Ошибка чтения файла
1055	Ошибка записи файла
1056	Ошибка закрытия файла
1069	Неизвестная ошибка файла
CSP (1200)	
1201	Ошибка контекста провайдера по OID
1202	Ошибка импорта открытого ключа
1203	Ошибка создания хэш
1204	Ошибка хэш данных
1205	Ошибка уничтожения хэш
1206	Ошибка проверки подписи
1207	Ошибка создания подписи или нет ключа подписи
1208	Ошибка проверки подписи
1209	Ошибка генерации ключа
1210	Ошибка экспорта открытого ключа
1211	Ошибка зашифровывания
1212	Ошибка расшифрования
1213	Ошибка импорта приватного ключа
1214	Ошибка экспорта приватного ключа
1215	Ошибка контекста сертификата
1216	Ошибка получения ключа
1217	Ошибка PIN-кода
1250	Несуществующий алгоритм
1299	Неизвестная ошибка провайдера
certificates (1300-1349)	
1301	Ошибка создания контекста сертификата
1302	Ошибка контекста свойств сертификата
1303	Ошибка релиза сертификата
1304	Ошибка поиска сертификата в хранилище
1305	Ошибка получения приватного ключа
1349	Неизвестная ошибка сертификата

Код ошибки	Описание
stores (1350-1399)	
1351	Ошибка открытия хранилища
1352	Ошибка закрытия хранилища
1353	Ошибка PFX хранилища
1354	Ошибка импорта PFX хранилища
1355	Ошибка добавления к PFX хранилищу
1399	Неизвестная ошибка хранилища
1400	Ошибка добавления сертификата в хранилище
ASN.1 (1500-1549)	
1501	Элемент отсутствует
1502	Идентификатор отсутствует
1503	Ошибка DER кода
1504	Значение отсутствует
1505	Ошибочный класс ASN структуры
1506	Ошибочное значение
1507	Ошибочный тег (элемент ASN структуры)
1508	Ошибка свойства тега
1509	Ошибка типа
1510	Ошибка синтаксиса ASN структуры
1511	Ошибка DER кодирования ASN структуры
1512	Элемент ASN структуры не пустой
1549	Неизвестная ошибка ASN структуры
BASE64(1550-1599)	
1551	Ошибка кодирования
1552	Ошибка декодирования
1599	Неизвестная ошибка
X509 (1600-1699)	
1601	Данные непригодны (недоступны)
Cryptography	
2001	Ошибка длины ключа
hashing (2100-2199)	
1552	Ошибка вычисления хэш (декодирования)
System Registry (-2010)	
2010	Нельзя открыть ключ реестра

Характерные Системные ошибки (winError)

Ошибка	Описание
2147023673	Действие было отменено пользователем
2147023901	Действие I/O было прервано из-за выхода из процесса или по запросу приложения
2147024816	Файл существует
2147418113	Катастрофическая ошибка
2146434962	Отменено пользователем
2146434965	PIN аутентификации токена неверный
2146435060	Нет смарт-карты

Ошибка	Описание
2146435066	SCARD нет памяти
2146435067	SCARD Ошибка устройства
2146881269	Неверное значение ASN.1 тега
2146885597	Неверная строка имени X500
2146885621	Сертификат не имеет приватного ключа
2146885628	Объект или свойство не найдено
2146893792	Внутренняя ошибка выполнения
2146889721	Неверное значение хэш
2146893795	Провайдер DLL устройства не был корректно инициализирован
2146893802	Контейнер ключа не существует или нет разрешения его открыть
2146893816	Задан неверный алгоритм
2146893821	Дефектный ключ
2147023673	Действие было отменено пользователем
2147023901	Действие I/O было прервано из-за выхода из процесса или по запросу приложения
2147024816	Файл существует
2147418113	Катастрофическая ошибка