



КАРМА версия 56.0.207

Руководство прикладного
программиста

©Электронные Офисные Системы. 2020 г.

Оглавление

1.	Назначение и описание интерфейса	4
2.	Описание функций интерфейсов	5
2.1.	Строка инициализации.....	5
2.2.	Общие параметры сертификата info	5
2.3.	Параметры сертификата certinfo	6
2.4.	Параметры расширения сертификата certextensioninf	6
2.5.	Параметры сертификата certinfo2	6
2.6.	Параметры расширения подписи extensioninfo	8
2.7.	Параметры цифровой подписи signinfo	8
2.8.	Формат адреса хранилища	9
2.9.	Списки адресов хранилищ	10
2.10.	Описание функций интерфейса EOCrypto.ICrypto	10
2.11.	Описание функций интерфейса EOCrypto.IPKI	19
2.12.	Описание HTTP интерфейса	24
3.	Рекомендации по использованию интерфейсов.....	26
3.1.	Создание объекта и инициализация интерфейса	26
3.2.	Функции шифрования	26
3.3.	Функции ЭЦП.....	26
3.4.	Функции работы с ключами и сертификатами.....	26
3.5.	Вспомогательные функции	27
3.6.	Замечания по многопоточной работе	27
4.	Пример использования интерфейса	28
4.1.	ActiveX: JavaScript для вычисления и проверки цифровой подписи	28
4.2.	ActiveX: JavaScript для шифрования и расшифрования файла	28
4.3.	ActiveX: JavaScript для использования события eventHandler	28
4.4.	Объявление интерфейса для подключения к событию eventHandler для C#.....	29
4.5.	ActiveX: JavaScript для составления списка из двух хранилищ.....	29
4.6.	XML HTTP Request	29
5.	CarmaCryptographySupport : использование криптографических функций, XmlDsig и XAdES в .Net.....	30
5.1.	Класс CarmaX509Certificate	30
5.2.	Класс CarmaX509Store	30
5.3.	Класс CarmaX509Chain	30
5.4.	Класс CarmaCMSSignatureContainer	30

5.5.	Класс CarmaCMSSignature	31
5.6.	Класс CarmaCMSSignatureExtension	31
5.7.	Класс CarmaCipher	31
5.8.	Класс CarmaCryptoServiceContext	31
5.9.	Класс CarmaSignedXml	31
5.10.	Класс XadesXml	31

1. Назначение и описание интерфейса

Интерфейс криптографического комплекса КАРМА предназначен для предоставления функций шифрования, установки и проверки ЭЦП, установки и проверки пользовательских сертификатов и вспомогательных функций через COM-интерфейсы, запросы по HTTP и NamedPipe протоколам.

Интерфейс состоит из набора динамических библиотек клиента, реализующих регистрируемые COM-интерфейсы [EOSCrypto.Icrypto](#) (Icrypto2 и Icrypto3) и [EOSPKILib.IPKI](#), библиотеки .Net CarmaCryptographySupport, скрипта carma.js и исполняемого модуля EosCryptoSvc, обеспечивающего взаимодействие со средствами криптографической защиты информации и электронно-цифровой подписи (СКЗИ), а также реализующего встроенный http – сервер. Взаимодействие COM – объектов, .Net клиента с исполняемым модулем осуществляется по http – подобному протоколу, базирующемуся на транспортном механизме NamedPipe. Взаимодействие с http – сервером осуществляется по стандартному протоколу HTTP с передачей данных в виде JSON.

2. Описание функций интерфейсов

В вызовах функций могут использоваться строки типа **string**, целочисленные значения **int**, ложь-истина типа **bool**, а также наборы (словари) пар ключ – значение типа **Scripting.Idictionary**. Словарь **Idictionary** – универсальный тип данных и может быть дополнен\изменен без изменения сигнатур методов, описание сущностей может отличаться в новых версиях.

2.1. Строка инициализации

Строка инициализации состоит из перечня параметров, разделенных символом «;» (точка с запятой). Каждый параметр – это пара имя-значение. Имя и значение разделены знаком «=» (равенство). Значение должно быть заключено в парные кавычки.

Параметр	Название	Описание значения
SERVER	адрес сервера КАРМА (в т.ч. сервер удалённой ¹ проверки ЭЦП)	В случае «localhost», «.», «127.0.0.1», если параметр не указан либо совпадает с именем/IP-адресом локального АРМ, то должен активизироваться локальный режим работы интерфейса Для инициализации NamedPipe канала необходимо указать адрес в формате: “pipe://<адрес компьютера>/carma” Для инициализации http соединения необходимо указать адрес в формате: “http://<адрес компьютера>:<порт>”
LOGMODE	Режим протоколирования	Допустимые значения: NO, ERRORS, FULL
LOGFILE	Имя файла протокола	Полный путь к файлу протокола.

2.2. Общие параметры сертификата **info**

Параметры сертификата пользователя **info** представляют собой словарь **Scripting.Idictionary**:

Ключ	Тип	Значение
SN	string	Серийный номер (значение поля serial сертификата открытого ключа)
CertName	string	Содержимое поля имени сертификата открытого ключа
Issuer	string	имя выдавшего центра (значение поля issuer сертификата открытого ключа).
ID	string	Идентификатор сертификата – результат конкатенации строк Serial и Issuer сертификата.

¹ В режиме удалённого использования доступны только функции: не требующие предъявления секретного ключа пользователя и вывода GUI. Все остальные функции, при попытке вызова, возвращают ошибку **ERROR Not supported** и при запросе возможности использования этих функций через функцию **CanDo** возвращается **ERROR Using remote server**.

2.3. Параметры сертификата **certinfo**

Параметры сертификата пользователя **certinfo** представляют собой словарь Scripting.Dictionary:

Ключ	Тип	Значение
Serial	string	Серийный номер
Issuer	string	Имя выдавшего центра сертификации (УЦ)
Description	string	содержимое поля имени сертификата открытого ключа
ValidFrom	string	Время начала действия сертификата по местному времени (времени компьютера)
ValidFromUTC	string	Время начала действия сертификата по Гринвичу (UTC)
ValidTo	string	Время окончания действия сертификата по местному времени (времени компьютера)
ValidToUTC	string	Время начала действия сертификата по Гринвичу (UTC)

2.4. Параметры расширения сертификата **certextensioninf**

Параметры расширения сертификата пользователя **certextensioninf** представляют собой словарь Scripting.Dictionary:

Ключ	Тип	Значение
OidFriendlyName	string	имя OID'а расширения (например, «Подписывание кода» для 1.3.6.1.5.5.7.3.3); при отсутствии такого в системе – значение OID
OidValue	string	OID расширения
RawData	string	закодированное в Base64 бинарное содержимое расширения
FormattedData	string	содержимое расширения, обработанное с помощью функции CryptFormatObject с флагом CRYPT_FORMAT_STR_MULTI_LINE
Critical	bool	Является ли расширение критическим

2.5. Параметры сертификата **certinfo2**

Параметры сертификата пользователя **certinfo2** представляют собой словарь Scripting.Dictionary:

Ключ	Тип	Значение
Serial	string	Серийный номер

Id	string	Идентификатор сертификата
Issuer	string	Имя выдавшего центра
Description	string	содержимое поля имени сертификата открытого ключа
Validity	string	Строковое представление состояния валидности сертификата
ValidityNumber	uint	Состояние валидности сертификата
HasPrivateKey	bool	Наличие или отсутствие связанного с сертификатом закрытого ключа
Version	string	Версия сертификата
SignatureID	string	Идентификатор алгоритма подписи
IssuerCert	IDictionary	certinfo2 сертификата УЦ, выдавшего данный сертификат.
OIDIssuer	string	Имя СА, выдавшего сертификат, в формате пар «OID=Значение», разделенных запятыми
OIDDescription	string	Имя Владельца в формате пар «OID=Значение», разделенных запятыми
X500Description	string	Имя Владельца в формате пар «X.500 name=Значение», разделенных запятыми
X500NameIssuer	string	Имя СА, выдавшего сертификат, в формате пар «X.500 name=Значение», разделенных запятыми
ValidFrom	string	Время начала действия сертификата по местному времени (времени компьютера)
ValidFromUTC	string	Время начала действия сертификата по Гринвичу (UTC)
ValidTo	string	Время окончания действия сертификата по местному времени (времени компьютера)
ValidToUTC	string	Время начала действия сертификата по Гринвичу (UTC)
PublicKeyAlgID	string	Идентификатор алгоритма открытого ключа
PublicKeyInfo	string	Дополнительная информация о ключе (если доступна)
IssuerUniqID	string	Уникальный номер СА, выдавшего сертификат
SubjectUniqID	string	Уникальный номер имени владельца сертификата
Extentions	IDictionary	Словарь содержащий расширения сертификата; ключ <i>ExtensionsCount</i> – Количество расширений сертификата; 2. <i>Extension1 ... ExtensionN</i> –

		Idispatch certextensioninfo – параметры расширения сертификата
RawData	string	Тело сертификата в формате Base64 строки
KeyUsage	uint	Использование ключа сертификата
KeyUsageStr	string	Строковое представление использования ключа сертификата

2.6. Параметры расширения подписи **extensioninfo**

Параметры расширения подписи **extensioninfo** представляют собой словарь Scripting.IDictionary:

Ключ	Тип	Значение
ExtensionName	string	Имя расширения
ExtensionOID	string	OID расширения
ExtensionInterpretedString	string	Строка-результат проверки. Используется для отображения в ShowSign
ExtensionInterpretedData	string	Данные-результат проверки

2.7. Параметры цифровой подписи **signinfo**

Параметры цифровой подписи **signinfo** представляют собой словарь Scripting.IDictionary:

Ключ	Тип	Значение
SignType	int	Тип подписи: 0 – отсоединенная подпись, 1 - присоединенная подпись
SignId	string	Идентификатор подписи внутри контейнера (сохранено в атрибуте "EOS signature parameters extension")
VerifyResult	string	Результат проверки подписи
Comment	string	Комментарий (сохранено в атрибуте "EOS signature parameters extension")
Uri	string	Идентификатор ресурса (сохранено в атрибуте "EOS signature parameters extension")
SignTime	string	Время создания подписи по местному времени (времени компьютера)
SignTimeUTC	string	Время создания подписи по Гринвичу (UTC)
SignAlg	string	OID алгоритма подписи
HashAlg	string	OID алгоритма хеширования

SignCert	IDictionary	certinfo сертификата подписи
CoSignCount	int	Количество соподписей (для соподписей данный параметр равен нулю)
CoSign1...CoSignN	IDictionary	signinfo соответствующей соподписи
ExtensionCount	int	Количество расширений
Extension1...ExtensionN	IDictionary	extensioninfo – параметры расширения

2.8. Формат адреса хранилища

Адрес хранилища представляет собой словарь IDictionary, который должен содержать следующие значения:

Ключ	Тип	Значение
StoreLocation	string	Идентификатор расположения хранилища (ID расположения)
StoreAddress	string	Адрес хранилища (Адрес)
StoreName	string	Имя хранилища (Имя)

Ниже приведена таблица задающая соответствие расположений и адресов хранилищ:

ID	(см. описание функции CertOpenStore)	Адрес
sscu	CERT_SYSTEM_STORE_CURRENT_USER	игнорируется
sscs	CERT_SYSTEM_STORE_CURRENT_SERVICE	игнорируется
sslm	CERT_SYSTEM_STORE_LOCAL_MACHINE	\\computer_name
sslmgp	CERT_SYSTEM_STORE_LOCAL_MACHINE_GROUP_POLICY	\\computer_name
sscugp	CERT_SYSTEM_STORE_CURRENT_USER_GROUP_POLICY	игнорируется
sss	CERT_SYSTEM_STORE_SERVICES	\\computer_name\service_name
ssu	CERT_SYSTEM_STORE_USERS	\\computer_name\user_name
sslme	CERT_SYSTEM_STORE_LOCAL_MACHINE_ENTERPRISE	игнорируется
remote	Удаленное хранилище	ldap://path/to/store, http://path/to/store при использовании данного расположения имя хранилища должно оставаться пустой строкой

Если словарь пуст или элемент **StoreLocation** представляет собой пустую строку, адрес хранилища считается неуказанным.

При некоторых вариантах расположения хранилища элементы **StoreName** или **StoreAddress** могут быть опущены.

Возможные примеры адресов хранилищ:

StoreLocation	StoreAddress	StoreName
“sscu”	отсутствует	“MY”

"sslm"	"\\HAL-9000"	"ROOT"
"ssu"	"\\srv1\User1"	"MY"
"remote"	"ldap://path/to/store"	отсутствует

2.9. Списки адресов хранилищ

Некоторые методы описанного ниже интерфейса [EOSPKILib.IPKI](#), позволяют принимать в качестве параметра список адресов хранилищ. Список адресов хранилищ представляет собой словарь IDictionary, структура которого является расширением словаря с единственным адресом и представлена ниже:

Ключ	Тип	Значение
StoreLocation	string	Идентификатор расположения хранилища (ID расположения)
StoreAddress	string	Адрес хранилища (Адрес)
StoreName	string	Имя хранилища (Имя)
AdditionalStores	IDictionary	Дополнительные адреса хранилищ Count - количество дополнительных адресов хранилищ; AdditionalStore1- AdditionalStoreN- IDictionary адреса дополнительного хранилища

2.10. Описание функций интерфейса EOSCrypto.ICrypto

Перечень функций интерфейса EOSCrypto.ICrypto их формальное описание представлено в следующей таблице:

Функция	Описание
string Init (string params)	Выполняет инициализацию компонента. Функция обязательно должна быть вызвана перед началом использования компонента. Функция может быть вызвана только один раз. params – строка содержащая список настроечных параметров. Функция возвращает "DONE" в случае успешной инициализации или "ERROR <ошибка>" при возникновении сбоя в процессе инициализации.

<p>string CanDo(string funcName)</p>	<p>Возвращает информацию о возможности выполнения операции funcName.</p> <p>Может быть использована для предварительной проверки доступности той или иной операции в данной реализации компонента при данной настройке. Имя операции должно соответствовать имени функции данного интерфейса, регистр символов не учитывается. Нельзя указывать в качестве параметра операции: “cando” и “init”.</p> <p>Возвращает “DONE” если функция доступна или “ERROR <причина>” в противном случае.</p>
<p>string SetCert(string certFilename, string certId)</p>	<p>Устанавливает в текущее хранилище ресурсы из файла certFilename и проверяет, что ресурсы соответствуют идентификатору certId. Файл должен быть в формате, совместимом с используемым провайдером.</p> <p>Например, в случае использования провайдеров, совместимых с CryptoAPI, файл может быть в форматах «cer» или «p7b». Возвращает “DONE”, “NEED <ресурс>” или “ERROR <ошибка>”.</p>
<p>string PreloadKey(string userCertId)</p>	<p>Создает контекст криптопровайдера и загружает в него закрытый ключ, соответствующий указанному сертификату. В зависимости от реализации и настроек допустимо несколько вариантов поведения: множественная предварительная загрузка – загрузка ключа в новый контекст, если ключ еще не был загружен; одинарная предварительная загрузка – загрузка ключа в текущий контекст с выгрузкой ранее загруженного ключа; однократная предварительная загрузка – если контекст еще не создан, то загрузка выполняется, в противном случае – команда игнорируется; отсутствие предварительной загрузки – функция не делает ничего; В случае, если по той или иной причине команда предварительной загрузки проигнорирована, то это не ошибочная ситуация, а предупреждение. Возвращает “DONE”, “NEED <ресурс>” либо “ERROR <ошибка>”. Результат – строка, описывающая выполненные действия. Например: ключ загружен, команда проигнорирована – ключ уже загружен, и т.п.</p>

<p>string Encrypt(string dataFilename, string senderCertId, string recipientCertId)</p>	<p>Шифрует содержимое файла dataFilename с использованием сертификата отправителя с идентификатором senderCertId и сертификата получателя с идентификатором recipientCertId. Зашифрованные данные сохраняются в файле с тем же именем, при этом исходные данные затираются. Возвращает “DONE”, “NEED <ресурс>” или “ERROR <ошибка>”.</p>
<p>string Decrypt(string dataFilename, string recipientCertId)</p>	<p>Расшифровывает содержимое файла dataFilename с использованием сертификата получателя с идентификатором recipientCertId. Расшифрованные данные сохраняются в файле с тем же именем, при этом исходные данные затираются. Возвращает “DONE”, “NEED <ресурс>” или “ERROR <ошибка>”.</p>
<p>string Sign(string dataFilename, string signFilename, string signatoryCerId)</p>	<p>Вычисляет ЭЦП для содержимого файла dataFilename и сохраняет ее в файле signFilename. Если файл signFilename существует, то он предварительно удаляется. При подписании в подпись включается вся цепочка сертификатов, необходимая для дальнейшей ее проверки. Возвращает “DONE”, “NEED <ресурс>” или “ERROR <ошибка>”.</p>
<p>string Verify(string dataFilename, string signFilename)</p>	<p>Проверяет цифровую подпись, содержащуюся в файле signFilename, на соответствие содержимому файла dataFilename. При значении аргумента signFilename равном “attached” предполагается, что dataFilename является контейнером присоединённой цифровой подписи. Возвращает “DONE”, “NEED <ресурс>” или “ERROR <ошибка>”. – может принимать значения “VALID” или “INVALID”. Если результат “VALID”, то в поле возвращается идентификатор сертификата подписанта. Если результат “INVALID”, то – список порядковых номеров (0 – первого уровня, 1..N- заверяющие) и описание причин, в связи с которыми подпись сочтена неверной. С помощью данной функции также осуществляется извлечение файла содержимого из присоединенной подписи. Для этого путь к самой подписи передается в параметре</p>

	<p>dataFilename, а в параметр signFilename передается строка вида attached" имя_файла". Результатом этой операции станет извлечение файла, содержащегося в присоединенной подписи в файл с именем имя_файла.</p>
<p>string GetCertInfo(string certFilename, Scripting.Dictionary info)</p>	<p>Читает файл сертификата certFilename, извлекает из него описательную информацию и помещает эту информацию в виде пар имя-значение в info. В результате, коллекция info обязательно должна содержать записи с именами «id» и «description». Объект коллекции должен быть создан на вызывающей стороне и передан в функцию как out (ref) параметр. Возвращает "DONE", "NEED <ресурс>" или "ERROR <ошибка>".</p>
<p>string IsKeyValid(string certFilename)</p>	<p>Читает сертификат или цепочку сертификатов из файла certFilename и проверяет действительность этого сертификата (цепочки сертификатов) на момент проверки. Возвращает "DONE ", "NEED <ресурс>" или "ERROR ". – одно из значений "VALID", "INVALID".</p>
<p>string ShowCert(string certFilename)</p>	<p>Показывает системное окно с информацией о ресурсе (обычно это сертификат) содержащемся в файле certFilename. Возвращает "DONE", "NEED <ресурс>" или "ERROR <ошибка>".</p>
<p>string TranslateCRL(string CRLFilename, string revokeCertIdFilename)</p>	<p>Читает файл CRLFilename, содержащий список отозванных сертификатов, извлекает из него серийные номера сертификатов и имена выдавших сертификаты центров и сохраняет список идентификаторов в файл revokeCertIdFilename. Разделителем серийного номера сертификата и имени выдавшего центра является символ #2 (ASCII 2) Разделителем пар является символ #1 (ASCII 1) Возвращает "DONE", "NEED <ресурс>" или "ERROR <ошибка>".</p>

Event NeedCert (eventArgs)	Если использующая система предоставляет функцию-обработчик данного события, то вместо возврата “NEED <ресурс>” все методы интерфейса будут вызывать данную функцию. Это событие реализовано с помощью механизма COM ConnectionPoint. Объект eventArgs имеет два свойства: certId – идентификатор требуемого ресурса – заполняется методом интерфейса перед вызовом обработчика события; certFilename – имя файла с требуемым ресурсом – заполняется обработчиком события.
Property eventHandler	Это свойство аналогично событию NeedCert. Добавлено для упрощения работы с событиями из скриптовых языков (например jscript). Реализовано с помощью вызова метода «по умолчанию» у диспинтерфейса.
string EncryptMulty (string dataFilename, string senderCertId, Scripting.Dictionary params)	Список полей params: (in) unsigned long recipientsCount (in) string recipientCertId1 ... (in) string recipientCertIdN Шифрует содержимое файла dataFilename с использованием сертификата отправителя с идентификатором senderCertId и сертификатов получателей с идентификаторами, перечисленными в recipientCertId(x). Алгоритм шифрования может быть задан в конфигурационном файле, если не задан или не подходит, то должен браться алгоритм по умолчанию для CSP. Зашифрованные данные сохраняются в файле с тем же именем, при этом исходные данные затираются. Значение useBase64 определяет, в какой кодировке будет записан результат операции: 0 – DER кодировка 1 – Base64 кодировка. Возвращает “DONE”, “NEED <ресурс>” или “ERROR <ошибка>”. В случае ошибки, функция возвращает HRESULT не равный S_OK (выбрасывает исключение в «бедных» средах) с присоединенным к нему ErrorInfo (сообщением об ошибке).
string Sign2 (string dataFilename, string signFilename, string signerCertId, bool Attached, unsigned int CertInclude, string	Вычисляет ЭЦП для содержимого файла dataFilename и сохраняет ее в файле signFilename. Если Attached=TRUE, то в signFilename формируется контейнер с присоединённой подписью. Значение CertInclude

<p>Comment, string Uri, bool AddSigner, bool useBase64)</p>	<p>определяет режим включения в состав подписи сертификата открытого ключа: 0 – сертификат не включается; 1 – в подпись включается конечный пользовательский сертификат; 2 – в подпись включается вся цепочка сертификатов для первой подписи и конечный пользовательский сертификат при добавлении в существующую подпись; Значение AddSigner определяет режим создания подписи: 0 – создание нового контейнера с подписью; 1 – добавление подписи в существующий контейнер Если файл signFilename существует, то он предварительно удаляется. Значение useBase64 определяет, в какой кодировке будет записан результат операции: 0 – DER кодировка 1 – Base64 кодировка. Возвращает “DONE”, “NEED <ресурс>” или “ERROR <ошибка>”. В случае ошибки, функция возвращает HRESULT не равный S_OK (выбрасывает исключение в «бедных» средах) с присоединенным к нему ErrorInfo (сообщением об ошибке)</p>
<p>string CoSign(string dataFilename, string signFilename, string signerCertId, unsigned int CertInclude, string Comment, string Uri, unsigned int SignerIndex)</p>	<p>Заверяет подпись из signFilename, исходный файл перезаписывается итоговым. При значении аргумента signFilename = “attached” предполагается, что dataFilename является контейнером присоединённой цифровой подписи. Значение CertInclude определяет режим включения в состав подписи сертификата открытого ключа: 0 – сертификат не включается; 1 – в подпись включается конечный пользовательский сертификат; 2 – в подпись включается цепочка сертификатов; Значение SignerIndex определяет, для какой из подписей контейнера делается соподпись; Возвращает “DONE”, “NEED <ресурс>” или “ERROR <ошибка>”. В случае ошибки, функция возвращает HRESULT не равный S_OK (выбрасывает исключение в «бедных» средах) с присоединенным к нему ErrorInfo (сообщением об ошибке)</p>

<p>string GetSignInfo(string dataFilename, string signFilename, Scripting.Dictionary params)</p>	<p>Список полей params: (out) unsigned long MainSignCount (out) iDispatch MainSign1 (signinfo) ... (out) iDispatch MainSignN (signinfo) Читает параметры ЭЦП (первого уровня и заверяющих, с распознаваемыми атрибутами), содержащейся в файле signFilename и помещает их в поля params. Возвращает "DONE", "NEED <ресурс>" или "ERROR <ошибка>". В случае ошибки, функция возвращает HRESULT не равный S_OK (выбрасывает исключение в «бедных» средах) с присоединенным к нему ErrorInfo (сообщением об ошибке).</p> <p>Для VBS доступна функция с поддержкой типа variant VBS_GetSignInfo.</p>
<p>string StopCryService(void)</p>	<p>Производит выгрузку криптографического сервиса. Для выполнения функции требуется предварительная инициализация. Возвращает "DONE" или "ERROR <ошибка>". В случае ошибки, функция возвращает HRESULT не равный S_OK (выбрасывает исключение в «бедных» средах) с присоединенным к нему ErrorInfo (сообщением об ошибке)</p>
<p>string ShowSign(string dataFilename, string signFilename)</p>	<p>Показывает окно информации о ЭЦП, содержащейся в файле signFilename, средствами криптоинтерфейса. Информация о ЭЦП должна содержать параметры signinfo, timestampinfo (при наличии), ocspinfo (при наличии), состояние валидности сертификата открытого ключа на текущий момент и результат проверки ЭЦП. Возвращает "DONE", "NEED <ресурс>" или "ERROR <ошибка>". В случае ошибки, функция возвращает HRESULT не равный S_OK (выбрасывает исключение в «бедных» средах) с присоединенным к нему ErrorInfo (сообщением об ошибке)</p>

<p>string Init2(string params)</p>	<p>Выполняет инициализацию компонента. Функция обязательно должна быть вызвана перед началом использования компонента. Функция может быть вызвана только один раз. params – строка содержащая список настроечных параметров. Функция возвращает “DONE” в случае успешной инициализации или “ERROR <ошибка>” при возникновении сбоя в процессе инициализации. В случае ошибки, функция возвращает HRESULT не равный S_OK (выбрасывает исключение в «бедных» средах) с присоединенным к нему ErrorInfo (сообщением об ошибке)</p>
<p>string Decrypt2(string dataFilename, string recipientCertId)</p>	<p>Расшифровывает содержимое файла dataFilename с использованием сертификата получателя с идентификатором recipientCertId. Расшифрованные данные сохраняются в файле с тем же именем, при этом исходные данные затираются. Возвращает “DONE”, “NEED <ресурс>” или “ERROR <ошибка>”. В случае ошибки, функция возвращает HRESULT не равный S_OK (выбрасывает исключение в «бедных» средах) с присоединенным к нему ErrorInfo (сообщением об ошибке)</p>
<p>string GetReceipientsList(string dataFilename, Scripting.Dictionary list)</p>	<p>Список полей params: (out) unsigned long ReceipientsCount (out) iDispatch ReceptientCertId1 (string) ... (out) iDispatch ReceptientCertIdN (string) Функция возвращает список идентификаторов сертификатов получателей зашифрованного контейнера, помещая их в поле list. Возвращает “DONE” или “ERROR <ошибка>”. В случае ошибки, функция возвращает HRESULT не равный S_OK (выбрасывает исключение в «бедных» средах) с присоединенным к нему ErrorInfo (сообщением об ошибке)</p> <p>Для VBS доступна функция с поддержкой типа variant VBS_GetReceipientsList.</p>

<p>string Hash(string dataFilename, string certId, Scripting.Dictionary params)</p>	<p>Возвращает хэш – функцию от данных в файле dataFilename. Хэш – алгоритм можно указать, используя идентификатор сертификата certId, либо явно указать OID или название алгоритма в словаре params по ключу HashAlgId. Названия некоторых алгоритмов:</p> <p>“gost-r-34.11-94” – ГОСТ Р 34.11</p> <p>“sha-1” – SHA-1</p> <p>“md5” – MD5</p> <p>“md2” – MD2</p> <p>“md4” – MD4</p>
<p>string SignHash(string dataFilename, string signFilename, string signerCertId, bool Attached, unsigned int CertInclude, string Comment, string Uri, bool AddSigner, bool useBase64)</p>	<p>Вычисляет ЭЦП для предварительно хешированных данных dataFilename и сохраняет ее в файле signFilename. Если Attached=TRUE, то в signFilename формируется контейнер с присоединённой подписью. Значение CertInclude определяет режим включения в состав подписи сертификата открытого ключа: 0 – сертификат не включается; 1 – в подпись включается конечный пользовательский сертификат; 2 – в подпись включается вся цепочка сертификатов для первой подписи и конечный пользовательский сертификат при добавлении в существующую подпись; Значение AddSigner определяет режим создания подписи: 0 – создание нового контейнера с подписью; 1 – добавление подписи в существующий контейнер Если файл signFilename существует, то он предварительно удаляется. Значение useBase64 определяет, в какой кодировке будет записан результат операции: 0 – DER кодировка 1 – Base64 кодировка. Возвращает “DONE”, “NEED <ресурс>” или “ERROR <ошибка>”. В случае ошибки, функция возвращает HRESULT не равный S_OK (выбрасывает исключение в «бедных» средах) с присоединенным к нему ErrorInfo (сообщением об ошибке)</p>

<pre>string DetachSignature(string attachedSignFileName, string detachedSignFileName, string dataFilename);</pre>	<p>Отсоединяет файл данных dataFilename и подпись detachedSignFileName из контейнера с присоединенной подписи attachedSignFileName. Возвращает "DONE" или "ERROR <ошибка>". В случае ошибки, функция возвращает HRESULT не равный S_OK (выбрасывает исключение в «бедных» средах) с присоединенным к нему ErrorInfo (сообщением об ошибке)</p>
<pre>string AttachSignature(string detachedSignFileName, string dataFilename, string attachedSignFileName);</pre>	<p>Присоединяет файл данных dataFilename к отсоединенной подписи detachedSignFileName в контейнер с присоединенной подписью attachedSignFileName. Возвращает "DONE" или "ERROR <ошибка>". В случае ошибки, функция возвращает HRESULT не равный S_OK (выбрасывает исключение в «бедных» средах) с присоединенным к нему ErrorInfo (сообщением об ошибке)</p>
<pre>string GetSignType(string signFileName);</pre>	<p>Возвращает тип подписи в файле signFileName. "attached" – присоединенная, "detached" – отсоединенная. Возвращает в случае ошибки "ERROR <ошибка>" и HRESULT не равный S_OK (выбрасывает исключение в «бедных» средах) с присоединенным к нему ErrorInfo (сообщением об ошибке)</p>

2.11. Описание функций интерфейса **EOCrypto.IPKI**

Перечень функций интерфейса EOCrypto.IPKI их формальное описание представлено в следующей таблице:

Функция	Описание
<pre>string Init(string params)</pre>	<p>Выполняет инициализацию компонента. Функция обязательно должна быть вызвана перед началом использования компонента. Функция может быть вызвана только один раз. params – строка содержащая список настроечных параметров. Функция возвращает "DONE" в случае успешной инициализации или "ERROR <ошибка>" при возникновении сбоя в процессе инициализации. В случае ошибки, функция возвращает HRESULT не равный S_OK (выбрасывает исключение в «бедных»</p>

	<p>средах) с присоединенным к нему <code>ErrorInfo</code> (сообщением об ошибке).</p>
<p>string CanDo(string funcName)</p>	<p>Возвращает информацию о возможности выполнения операции <code>funcName</code>.</p> <p>Может быть использована для предварительной проверки доступности той или иной операции в данной реализации компонента при данной настройке. Имя операции должно соответствовать имени функции данного интерфейса, регистр символов не учитывается. Нельзя указывать в качестве параметра операции: “cando” и “init”.</p> <p>Возвращает “DONE” если функция доступна или “ERROR <причина>” в противном случае.</p> <p>В случае ошибки, функция возвращает <code>HRESULT</code> не равный <code>S_OK</code> (выбрасывает исключение в «бедных» средах) с присоединенным к нему <code>ErrorInfo</code> (сообщением об ошибке).</p>
<p>string SetCurrentStore (Scripting.Dictionary params)</p>	<p>Выполняет установку рабочего хранилища.</p> <p>Поддерживает работу со списками хранилищ. В случае получения списка в качестве рабочего хранилища устанавливается коллекция хранилищ из списка.</p> <p>Список элементов <code>params</code>: (in) <code>params</code> – словарь, содержащий хранилища (см. Формат адреса хранилища) либо список адресов хранилищ. По умолчанию рабочим хранилищем является пользовательской личное хранилище сертификатов (MY). Функция возвращает “DONE” в случае успеха или “ERROR <ошибка>” при возникновении ошибки. В случае ошибки, функция возвращает <code>HRESULT</code> не равный <code>S_OK</code> (выбрасывает исключение в «бедных» средах) с присоединенным к нему <code>ErrorInfo</code> (сообщением об ошибке).</p>

<p>string GetCurrentStore (Scripting.Dictionary params)</p>	<p>Возвращает информацию о перечне хранилищ сертификатов открытых ключей установленных по умолчанию. Список элементов params: (in) params – словарь, содержащий хранилища (см. Формат адреса хранилища) либо список адресов хранилищ. Функция возвращает “DONE” в случае успеха или “ERROR <ошибка>” при возникновении ошибки. В случае ошибки, функция возвращает HRESULT не равный S_OK (выбрасывает исключение в «бедных» средах) с присоединенным к нему ErrorInfo (сообщением об ошибке).</p> <p>Для VBS доступна функция с поддержкой типа variant VBS_GetCurrentStore.</p>
<p>string EnumStore (Scripting.Dictionary params)</p>	<p>Возвращает информацию о перечне хранилищ сертификатов открытых ключей по указанному адресу. Список полей params: (in) первые три элемента словаря должны содержать адрес хранилища (см. Формат адреса хранилища) (out) unsigned long StoreCount (out) string StoreName1 ... (out) string StoreName StoreCount</p> <p>После завершения работы функция дополняет словарь именами хранилищ. Если адрес хранилища не указан, функция возвращает список доступных хранилищ в текущем пользовательском контексте. Объект коллекции должен быть создан на вызывающей стороне и передан в функцию как out (ref) параметр. ВНИМАНИЕ! Удаленные хранилища (remote) не поддерживаются. Функция возвращает “DONE” в случае успеха или “ERROR <ошибка>” при возникновении ошибки. В случае ошибки, функция возвращает HRESULT не равный S_OK (выбрасывает исключение в «бедных» средах) с присоединенным к нему ErrorInfo (сообщением об ошибке).</p> <p>Для VBS доступна функция с поддержкой типа variant VBS_EnumStore.</p>
<p>string EnumCertificates (Scripting.Dictionary params)</p>	<p>Возвращает информацию о перечне сертификатов открытых ключей, доступных в хранилище по указанному адресу. Список полей params: (in) первые три элемента словаря должны содержать адрес</p>

	<p>хранилища (см. Формат адреса хранилища) (out) unsigned long certCount (out) string certID1 ... (out) string certID certCount После завершения работы функция дополняет словарь идентификаторами сертификатов хранилищ. Если адрес хранилища не указан, функция возвращает список доступных сертификатов в пользовательском личном хранилище сертификатов (MY). Объект коллекции должен быть создан на вызывающей стороне и передан в функцию как out (ref) параметр. Функция возвращает "DONE" в случае успеха или "ERROR <ошибка>" при возникновении ошибки. В случае ошибки, функция возвращает HRESULT не равный S_OK (выбрасывает исключение в «бедных» средах) с присоединенным к нему ErrorInfo (сообщением об ошибке).</p> <p>Для VBS доступна функция с поддержкой типа variant VBS_EnumCertificates.</p>
<p>string GetCertInfo2(string certID, Scripting.Dictionary certinfo2)</p>	<p>Возвращает параметры certinfo2 сертификата с идентификатором certID. Поиск сертификата выполняется в текущем рабочем хранилище сертификатов. Объект коллекции должен быть создан на вызывающей стороне и передан в функцию как out (ref) параметр. Функция возвращает "DONE" в случае успеха или "ERROR <ошибка>" при возникновении ошибки. В случае ошибки, функция возвращает HRESULT не равный S_OK (выбрасывает исключение в «бедных» средах) с присоединенным к нему ErrorInfo (сообщением об ошибке).</p> <p>Для VBS доступна функция с поддержкой типа variant VBS_GetCertInfo2.</p>
<p>string GetCertInfo3(string certID, bool checkValidity, Scripting.Dictionary certinfo2)</p>	<p>Аналогично вызову GetCertInfo2 опционально без проверки сертификата при checkValidity = false Для VBS доступна функция с поддержкой типа variant VBS_GetCertInfo3.</p>
<p>string GetCertInfo3Multy(string certIDList, bool checkValidity, Scripting.Dictionary certinfo2)</p>	<p>Аналогично вызову GetCertInfo3 CertIDList может принимать строку с перечислением идентификаторов сертификатов. Для VBS доступна функция с поддержкой типа variant VBS_GetCertInfo3Multy.</p>

<p>string GetCert(string certID, string certFilename)</p>	<p>Получает содержимое сертификата с идентификатором certID и сохраняет его в файл с именем certFilename. Поиск сертификата выполняется в текущем рабочем хранилище сертификатов. Функция возвращает “DONE” в случае успеха или “ERROR <ошибка>” при возникновении ошибки. В случае ошибки, функция возвращает HRESULT не равный S_OK (выбрасывает исключение в «бедных» средах) с присоединенным к нему ErrorInfo (сообщением об ошибке).</p>
<p>string ShowCert2(string certID)</p>	<p>Показывает окно с информацией о сертификате с идентификатором certID. Поиск сертификата выполняется в текущем рабочем хранилище сертификатов. Функция возвращает “DONE” в случае успеха или “ERROR <ошибка>” при возникновении ошибки. В случае ошибки, функция возвращает HRESULT не равный S_OK (выбрасывает исключение в «бедных» средах) с присоединенным к нему ErrorInfo (сообщением об ошибке).</p>
<p>string StopCryService(void)</p>	<p>Производит выгрузку криптографического сервиса. Для выполнения функции требуется предварительная инициализация. Возвращает “DONE” или “ERROR <ошибка>”. В случае ошибки, функция возвращает HRESULT не равный S_OK (выбрасывает исключение в «бедных» средах) с присоединенным к нему ErrorInfo (сообщением об ошибке)</p>
<p>string TranslateCRL2(string CRLID, Scripting.Dictionary revokeCertIdList)</p>	<p>Читает CRL с идентификатором CRLID, извлекает из него идентификаторы сертификатов (серийные номера сертификатов и имена выдавших сертификаты центров) и возвращает список идентификаторов в revokeCertIdList. Список полей revokeCertIdList: (out) unsigned long CertIdCount (out) string CertId1 ... (out) string CertId CertIdCount Поиск CRL выполняется в текущем рабочем хранилище сертификатов. Объект коллекции должен быть создан на вызывающей стороне и передан в функцию как out (ref) параметр. Функция возвращает “DONE” в случае успеха или “ERROR <ошибка>” при возникновении ошибки. В случае ошибки, функция возвращает HRESULT не</p>

	<p>равный S_OK (выбрасывает исключение в «бедных» средах) с присоединенным к нему ErrorInfo (сообщением об ошибке).</p> <p>Для VBS доступна функция с поддержкой типа variant VBS_TranslateCRL2.</p>
<p>string SetCert2(string CertFilename, Scripting.Dictionary certInfo)</p>	<p>Загружает во временное хранилище указанный файл, содержащий сертификаты. Файл может быть любого из поддерживаемых ОС Windows формата. При передаче в функцию certInfo, возвращает в нём информацию об установленных сертификатах. Список полей certInfo: (out) unsigned long CertInfoCount (out) Scripting.Dictionary CertInfo1 - содержит certinfo2 для данного сертификата ... (out) Scripting.Dictionary CertInfoCertCount</p> <p>Для VBS доступна функция с поддержкой типа variant VBS_SetCert2.</p>
<p>string SetCertWithPrivateKey(string CertFilename, string certFilePassword, Scripting.Dictionary certInfo)</p>	<p>Загружает во временное хранилище указанный файл, содержащий сертификаты с закрытым ключом. Файл может быть любого из поддерживаемых ОС Windows формата. При передаче в функцию certInfo, возвращает в нём информацию об установленных сертификатах. Список полей certInfo: (out) unsigned long CertInfoCount (out) Scripting.Dictionary CertInfo1 - содержит certinfo2 для данного сертификата ... (out) Scripting.Dictionary CertInfoCertCount</p> <p>Для VBS доступна функция с поддержкой типа variant VBS_SetCertWithPrivateKey.</p>

2.12. Описание HTTP интерфейса

Доступ к сервису KAPMA может быть осуществлен по HTTP – протоколу. В этом случае обмен данными происходит в виде JSON. HTTP сервер KAPMA может работать как на локальном, так и на удаленном компьютере. Таким образом, можно реализовать работу с криптографией из любого браузера на клиентской стороне с доступом к секретным ключам. Кроме этого, COM – объекты EosCrypto и EosPKI могут быть инициализированы на работу по HTTP – протоколу (вместо Named pipes по умолчанию), для этого необходимо указать в параметрах инициализации SERVER="http://<адрес:порт HTTP сервера KAPMA>" (например, "http://localhost:8080").

Пример JSON данных запроса:

```
{
  "mode":24,
  "currentStores":[{"location":"sscu","address":"","name":"MY"}],
```

```
"extInitParams": "SERVER=\"http://localhost:8080\""
```

}
 “mode” – указывает, какая функция запрашивается (список значений есть в файле carma.js),
 “stores” – список активных хранилищ,
 “extInitParams” – дополнительные параметры инициализации.

Пример ответа сервиса:

```
{
  "errorCode": 0,
  "errorMessage": "DONE"
}
```

“errorCode” – код ошибки (для s_ok равен 0)
 “errorMessage” – сообщение об ошибке, в случае s_ok «DONE»

Для **JavaScript** поставляется скрипт – файл **carma.js** (<http://localhost:8080/carma.js>). В нем описан интерфейс **CarmaConnectionInterface** и его реализации: **CarmaHttp** и **CarmaActiveX**. Унифицированный интерфейс **CarmaConnectionInterface** позволяет взаимодействовать с сервисом КАРМА как через COM (для IE), так и HTTP (Chrome, FF, Safari и т.п. браузеры без поддержки технологии ActiveX).

Файл **carma.js** находится в подпапке Web директории установки.

3. Рекомендации по использованию интерфейсов

В составе установки системы КАРМА поставляется расширение для контекстного меню проводника. Данное расширение написано на языке C# и может быть использовано как пример встраивания системы. Исходный код расширения может быть выслан по запросу в техническую поддержку.

3.1. Создание объекта и инициализация интерфейса

После инсталляции интерфейса становится доступной процедура создания объекта (здесь и далее будут представлены примеры для java script и C#)

```
var EOSCryptoActivex = new ActiveXObject("EosUtils.EosCryptoSvc");
```

перед использованием функций интерфейса необходимо выполнить процедуру инициализации:

```
var init = EOSCryptoActivex.Init(connstring);
```

пример строки соединения:

```
var connstring =
"SERVER=\"localhost\";LOGFILE=\"c:\\EOSCrypto.log\";LOGMODE=\"FULL\""
```

3.2. Функции шифрования

Здесь и далее переменные показаны для примера подготовки к вызову функций интерфейса:

```
var datafile = "crtest.txt"
var sendercertid = "61EF6D07000000000006TEST CA"
var recepientscertid = "72EF7D08000000000007TEST CA"
```

Шифрование:

```
var encrypt = EOSCryptoActivex.Encrypt(datafile, sendercertid,
recepientscertid);
```

Расшифрование:

```
var decrypt = EOSCryptoActivex.Decrypt(datafile, recepientscertid);
```

3.3. Функции ЭЦП

```
var signfile = "crtest_sign.txt"
```

Вычисление цифровой подписи:

```
var sign = EOSCryptoActivex.Sign(datafile, signfile,certid);
```

Проверка цифровой подписи:

```
var verify = EOSCryptoActivex.Verify(datafile, signfile);
```

3.4. Функции работы с ключами и сертификатами

```
var certid = "61EF6D07000000000006TEST CA"
var cert = "certificate.cer"
var certinfo = new ActiveXObject("Scripting.Dictionary");
```

Кэширование ключа пользователя:

```
var preloadkey = EOSCryptoActivex.PreloadKey(certid);
```

Проверка валидности сертификата:

```
var iskeyvalid = EOSCryptoActivex.IsKeyValid(cert);
```

Получение значений полей сертификата в объект Dictionary:

```
var getcinfo = EOSCryptoActivex.GetCertInfo(cert, certinfo);
```

Показ сертификата:

```
var showcert = EOSCryptoActivex.ShowCert(cert);
```

Установка сертификата из файла в хранилище:

```
var setcert = EOSCryptoActivex.SetCert(cert, certid);
```

3.5. Вспомогательные функции

Проверка возможности выполнения функции «PreloadKey» в данной конфигурации интерфейса:

```
var cando = EOSCryptoActivex.CanDo("PreloadKey");
```

Преобразование [CRL](#) в текстовый файл:

```
var crl = "certcrl.crl"
var crltext = "crltext.txt"
var getcrl = EOSCryptoActivex.TranslateCRL(crl, crlout);
```

3.6. Замечания по многопоточной работе

Интерфейсы EOSCrypto.ICrypto и EOSCrypto.IPKI взаимосвязаны списком текущих хранилищ сертификатов (вызов PKI.SetCurrentStore влияет на Crypto2). При многопоточной работе могут быть проблемы в окружении .Net CLR, и для полной изоляции пар COM - интерфейсов необходимо передать идентификатор клиента – уникальную строку в пределах процесса. Для этого в строку инициализации EOSCrypto.ICrypto и EOSCrypto.IPKI необходимо добавить параметр CLIENT_ID="<идентификатор клиента>" .

Например:

```
CLIENT_ID="\24CDE36A-806E-4671-A4F6-28A6B28EA1BC\";
```

4. Пример использования интерфейса

4.1. ActiveX: JavaScript для вычисления и проверки цифровой подписи

```

var cert = "certificate.cer"
var certid = "61EF6D0700000000000006TEST CA"
var datafile = "crtest.txt"
var signfile = "crtest_sign.txt"
var connstring =
"SERVER=\"localhost\";LOGFILE=\"c:\\EOSCrypto.log\";LOGMODE=\"ERRORS\""
  var EOSCryptoActivex = new ActiveXObject("EosUtils.EosCryptoSvc");
var init = EOSCryptoActivex.Init(connstring);
var preloadkey = EOSCryptoActivex.PreloadKey(certid);
var setcert = EOSCryptoActivex.SetCert(cert, certid);
var sign = EOSCryptoActivex.Sign(datafile, signfile, certid);
Echo("Signing result "+ sign);

var verify = EOSCryptoActivex.Verify(datafile, signfile);
Echo("Verify result "+ verify);

```

4.2. ActiveX: JavaScript для шифрования и расшифрования файла

```

var sendercert = "scertificate.cer"
var receipientcert = "rcertificate.cer"
var sendercertid = "61EF6D0700000000000006TEST CA"
var receipientcertid = "72EF7D0800000000000007TEST CA"
var datafile = "crtest.txt"
var connstring = "SERVER=\"localhost\";LOGMODE=\"OFF\""

var EOSCryptoActivex = new ActiveXObject("EosUtils.EosCryptoSvc");
var init = EOSCryptoActivex.Init(connstring);
var preloadkey = EOSCryptoActivex.PreloadKey(sendercertid);
var setcert = EOSCryptoActivex.SetCert(sendercert, sendercertid);
var setcert = EOSCryptoActivex.SetCert(receipientcert, receipientcertid);
var encrypt = EOSCryptoActivex.Encrypt(datafile, sendercertid,
receipientcertid);
Echo("Encrypting result "+ encrypt);

var decrypt = EOSCryptoActivex.Decrypt(datafile, receipientcertid);
Echo("Decrypting result "+ decrypt);

```

4.3. ActiveX: JavaScript для использования события eventHandler

```

function OnNeedCert(args)
{
  ...
  //Используем переданный args.certId для получения нужного ресурса
  ...
  args.certFilename = "certificate";
  return;
} ...
var crypto = CreateActiveXObject("EosUtils.EosCryptoSvc");
crypto.eventHandler = OnNeedCert;
...

```

4.4. Объявление интерфейса для подключения к событию `eventHandler` для C#

```
[InterfaceType(ComInterfaceType.InterfaceIsIDispatch)]
public interface ICryptoEventHandler
{
    [DispId(0)]
    void OnNeedCert(INeedEventArgs eventArgs);
}
[ClassInterface(ClassInterfaceType.None)]
public class CryptoEventHandler : ICryptoEventHandler
{
    public void OnNeedCert(EOSCrypto.INeedEventArgs args)
    {
        args.certFilename = @"certificate.cer";
    }
}
```

4.5. ActiveX: JavaScript для составления списка из двух хранилищ

```
mainDict.add("StoreName", "MY");
mainDict.add("StoreLocation", "sscu");

addDict1.add("StoreLocation", "sslm");
addDict1.add("StoreAddress", "computer.local");
addDict1.add("StoreName", "ROOT");

addDicts.add("Count", 1);
addDicts.add("AdditionalStore1", addDict1);
mainDict.add("AdditionalStores", addDicts);
```

4.6. XML HTTP Request

Так как сервис КАРМА оснащен полноценным http сервером, то возможно прямо из браузера выполнить запросы, не используя прослойки COM. С примерами запросов можно ознакомиться на странице <http://localhost:8080/test>

5. **CarmaCryptographySupport**: использование криптографических функций, **XmlDsig** и **XAdES** в .Net

CarmaCryptographySupport является сборкой .Net (версии CLR 2.0 и 4.0, Core 3.1), которая использует крипто-интерфейсы КАРМА и реализует работу с криптографическими данными, XmlDsig и XAdES подписями с поддержкой ГОСТ алгоритмов. Работа с сертификатами, подписями CMS ([Cryptography Message Syntax](#)), шифрование осуществлены через классы – «обертки», которые упрощают прикладной код и используют удобства среды .Net и языка C# в частности. В зависимости от версии библиотеки (3.0 \ 4.0) для взаимодействия используется либо COM – объекты, либо прямые запросы Http\NamedPipe. Также есть версия библиотеки для .Net Core.

Основные классы:

- CarmaX509Certificate
- CarmaX509Store
- CarmaX509Chain
- CarmaCMSSignatureContainer
- CarmaCMSSignature
- CarmaCipher
- CarmaCryptoServiceContext
- CarmaCMSSignatureExtension
- CarmaSignedXml
- XadesSignedXml

5.1. Класс **CarmaX509Certificate**

CarmaX509Certificate представляет сертификат. Является оберткой для [certinfo2](#).

Некоторые свойства и методы совпадают с

System.Security.Cryptography.X509Certificates.X509Certificate2.

Пример использования доступен в проекте CertificateTestApp решения CCSDotNetTestApps.

5.2. Класс **CarmaX509Store**

CarmaX509Store представляет хранилище с сертификатами.

Пример использования доступен в проекте CertificateTestApp решения CCSDotNetTestApps.

5.3. Класс **CarmaX509Chain**

CarmaX509Chain представляет цепочку сертификатов.

Пример использования доступен в проекте CertificateTestApp решения CCSDotNetTestApps.

5.4. Класс **CarmaCMSSignatureContainer**

CarmaCMSSignatureContainer представляет сообщение Cryptography Message Syntax (далее CMS) в формате ASN.1 файла. С помощью данного класса можно открыть файл и получить

данные всех подписей. Также доступны методы для добавления подписи и соподписи в контейнер.

Пример использования доступен в проекте CMSTestApp решения CCSDotNetTestApps.

5.5. Класс **CarmaCMSSignature**

CarmaCMSSignature представляет подпись CMS. Является оберткой для [signinfo](#). Текущая версия КАРМА не предоставляет возможность работы напрямую с данными криптографического сообщения CMS.

Пример использования доступен в проекте CMSTestApp решения CCSDotNetTestApps.

5.6. Класс **CarmaCMSSignatureExtension**

CarmaCMSSignatureExtension представляет расширение подписи CMS. Является оберткой для [extensioninfo](#). Заранее известны 2 расширения:

- Штамп времени **CarmaCMSTimeStampExtension**
- Графическая подпись (факсимиле) **CarmaCMSGraphSignExtension**

Пример использования доступен в проекте CMSTestApp решения CCSDotNetTestApps.

5.7. Класс **CarmaCipher**

CarmaCipher предоставляет функции шифрования данных.

Пример использования доступен в проекте CipherTestApp решения CCSDotNetTestApps.

5.8. Класс **CarmaCryptoServiceContext**

CarmaCryptoServiceContext представляет контекст экземпляров COM-интерфейсов либо Http\NamedPipe запросов. Данный класс создает и инициализирует COM – интерфейсы, устанавливает возможность соединения с сервисом КАРМА. Кроме этого, данный класс предназначен для управления временными файлами, необходимыми в работе библиотеки. Для корректной многопоточной работы необходимо использовать отдельный экземпляр **CarmaCryptoServiceContext** на поток. Экземпляр CarmaCryptoServiceContext можно использовать в качестве аргумента конструктора крипто-классов в **CarmaCryptographySupport**. По умолчанию используется статический экземпляр DefaultInstance.

5.9. Класс **CarmaSignedXml**

CarmaSignedXml представляет оболочку основного объекта подписи XML для упрощения создания подписей [XmlDsig](#) с поддержкой ГОСТ алгоритмов. Внутренняя структура и сопутствующие классы переключаются с System.Security.Cryptography.Xml.SignedXml

5.10. Класс **XadesXml**

XadesXml является наследованным классом [CarmaSignedXml](#) и представляет оболочку основного объекта подписи [XML Advanced Electronic Signatures \(XAdES\)](#) – улучшенной подписи XML с поддержкой ГОСТ алгоритмов.

Пример использования доступен в проекте XadesTestApp решения CCSDotNetTestApps.