Код Безопасности

Программно-аппаратный комплекс
квалифицированной электронной подписи

"Jinn". Версия 1.0

Руководство программиста

RU.88338853.501430.008 33

Почтовый адрес:	**115127, Россия, Москва, а/я 66** **ООО "Код Безопасности"**
Телефон:	**8 495 982-30-20**
E-mail:	**info@securitycode.ru**
Web:	**http://www.securitycode.ru**

Оглавление

Введение ... **4**

Общие сведения ... **5**

Назначение и основные функции ... 5

Системные требования ... 5

Структура "Jinn-Client" ... 6

Средства электронной подписи ... 6

Модуль JBP ... 7

Использование "Jinn-Client" ... **9**

Настройка отображения заголовков в окне визуализации "Jinn-Client" ... 9

Синтаксис заголовка ... 9

Допустимые значения полей ... 9

Проверка программы ... 11

Методы и свойства объекта "Jinn-Client" ... 12

Метод CreateXmlCryptoContext ... 12

Метод CreateBinaryBase64CryptoContext ... 13

Пример реализации объекта "Jinn-Client" ... 14

Методы и свойства объекта плагина ... 16

Метод CreateXmlCryptoContext ... 16

Метод CreateBinaryBase64CryptoContext ... 17

Метод CreateXmlCryptoContextAsync ... 18

Метод CreateBinaryBase64CryptoContextAsync ... 19

Свойство LastError ... 21

Пример реализации объекта плагина ... 21

Плагин JinnSignExtensionProvider ... 23

Описание класса CryptoContextXML ... 24

Описание класса CryptoContextBinaryBase64 ... 25

Пример взаимодействия ... 27

Сообщения ПАК "Jinn" ... **28**

Сообщения и ошибки уровня бизнес-логики ... 28

Пример обработки исключений в JavaScript ... 34

Введение

Данное руководство предназначено для программистов, использующих изделие "Программно-аппаратный комплекс квалифицированной электронной подписи "Jinn". Версия 1.0" RU.88338853.501430.008 (далее — ПАК "Jinn", изделие, комплекс). В нем содержатся сведения об использовании функций API (application programming interface) ПАК "Jinn".

Условные обозначения

В руководстве для выделения некоторых элементов текста используется ряд условных обозначений.

Внутренние ссылки обычно содержат указание на номер страницы с нужными сведениями. Ссылки на другие документы или источники информации размещаются в тексте примечаний или на полях.

Важная и дополнительная информация оформлена в виде примечаний. Степень важности содержащихся в них сведений отображают пиктограммы на полях.

- Так обозначается дополнительная информация, которая может содержать примеры, ссылки на другие документы или другие части этого руководства.

- Такой пиктограммой выделяется важная информация, которую необходимо принять во внимание.

- Эта пиктограмма сопровождает информацию предостерегающего характера.

Исключения. Примечания могут не сопровождаться пиктограммами. А на полях, помимо пиктограмм примечаний, могут быть приведены и другие графические элементы, например, изображения кнопок, действия с которыми упомянуты в тексте расположенного рядом абзаца.

Другие источники информации

Сайт в Интернете. Если у вас есть доступ в Интернет, вы можете посетить сайт компании "Код Безопасности" (http://www.securitycode.ru/) или связаться с представителями компании по электронной почте (support@securitycode.ru).

Учебные курсы. Освоить аппаратные и программные продукты компании "Код Безопасности" можно в авторизованных учебных центрах. Перечень учебных центров и условия обучения представлены на сайте компании http://www.securitycode.ru/company/education/training-courses/. Связаться с представителем компании по вопросам организации обучения можно по электронной почте (education@securitycode.ru).

Глава 1
Общие сведения

В корпоративных, территориально распределенных информационных системах могут циркулировать электронные документы, требующие заверения электронной подписью. Существует и признается актуальной атака "человек посередине". Этим человеком может быть физическое лицо, непосредственно взаимодействующее с компьютером ответственного за работу с документом лица, или удаленный злоумышленник, который действует посредством программной атаки (всевозможные вирусы, трояны, черви, руткиты и т. д.) на компьютер ответственного лица из глобальной или локальной сети. Если с первым типом злоумышленника (физическое лицо) можно бороться только организационными мерами, то со вторым необходимо бороться программными или программно-аппаратными средствами. Для защиты именно от второго типа злоумышленника предлагается решение ПАК "Jinn".

Назначение и основные функции

ПАК "Jinn" предназначен для формирования электронной подписи (ЭП) электронного документа, расположенного в ОЗУ компьютера в виде XML-документа, текстового или бинарного файла. Формирование ЭП осуществляется в соответствии с положениями ст. 12 Федерального закона РФ "Об электронной подписи" от 06.04.11 № 63-ФЗ.

ПАК "Jinn" реализует следующие основные функции:

- визуализация электронного документа при отображении подписываемого или проверяемого электронного документа;
- формирование электронной подписи в соответствии с ГОСТ Р 34.10–2001, ГОСТ Р 34.11–94, ГОСТ Р 34.10–2012, ГОСТ Р 34.11–2012;
- генерация ключей электронной подписи и формирование запросов на создание сертификатов.

Для проверки правильности работы алгоритма выработки ЭП в комплексе реализована вспомогательная функция контрольного тестирования. Данная функция не используется в качестве целевой функции проверки ЭП.

ПАК "Jinn" можно эксплуатировать совместно со следующими программными продуктами (для файлов с расширениями .txt, .odt, .xml, .pdf) без проведения тематических исследований и/или оценки влияния:

- Adobe Acrobat Reader (от версии 11.0 и выше);
- MS Word 2007, Word 2010, Word 2013.

Системные требования

Компьютеры, на которых предполагается использовать ПАК "Jinn", должны соответствовать следующим аппаратным и программным требованиям:

Операционная система	MS Windows 10, 8.1 x86/x64, 8 x86/x64, 7 SP1 x86/x64, Vista SP2 x86/x64 (кроме всех выпусков Starter); MS Windows XP Professional SP3 x86/x64; MS Windows 2008 Server SP2 x86/x64, 2008 Server R2 SP1 x64, 2003 Server R2 SP2 x86/x64, 2003 Server SP2 x86/x64
Процессор при использовании режима эмуляции доверенной среды (ДС) в ОС (исполнения 7, 8, 9 и 3–6 в режиме эмуляции ДС в ОС)	В соответствии с требованиями ОС, установленной на компьютер

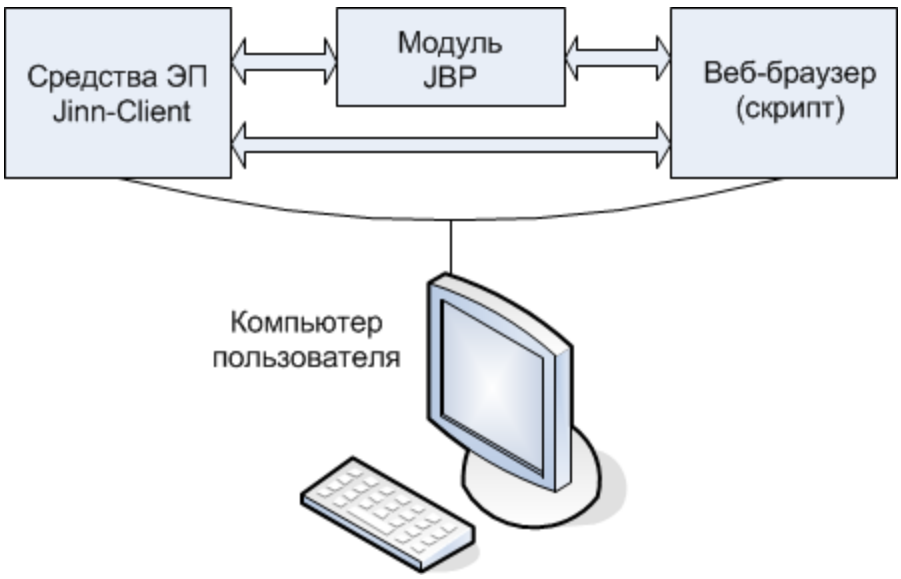
Процессор при использовании режима ДС (исполнения 1,2 и 3–6 в режиме ДС)	Многоядерный (2 и более) процессор Intel должен поддерживать технологии Intel-VT (VT-x) и EPT. Многоядерный (2 и более) процессор AMD должен поддерживать технологию AMD-V
Оперативная память	В соответствии с требованиями ОС, установленной на компьютер
Жесткий диск (свободное место)	50 МБ
Привод	Привод DVD/CD-ROM
Интерфейсы	2 x USB 2.0; 1 x PCI-E — для исполнений 3, 4, 7, 8, 9
Дополнительное ПО	ПАК "Соболь" — для исполнений 3, 4, 7, 8

- Доверенная среда несовместима с энергосберегающим режимом "Гибернация" ОС Windows.
- На компьютерах с материнскими платами производителя ASUS, использующих некоторые старые версии BIOS, возможны проблемы с загрузкой доверенной среды с USB-флеш-накопителей, подключенных к порту USB 3.0. Для устранения таких проблем рекомендуется обновить BIOS до последней версии. Также можно использовать имеющийся в BIOS режим загрузки компьютера по нажатии клавиши F8 или в настройках загрузки BIOS указывать конкретное устройство, с которого выполняется загрузка.

Структура "Jinn-Client"

Программное обеспечение (ПО) "Jinn-Client" устанавливается на рабочее место пользователя и используется для создания ЭП документов. "Jinn-Client" содержит в своем составе следующие компоненты:

- средства электронной подписи "Jinn-Client";
- модуль Jinn Browser Plugin (JBP) для взаимодействия средств ЭП с веб-браузером.



Средства электронной подписи

В комплексе ПАК "Jinn" функции формирования ЭП документа, контрольного тестирования (проверки) ЭП, визуализации подписываемого документа реализуются с помощью программного компонента "Jinn-Client". Программное обеспечение "Jinn-Client" устанавливается администратором на компьютер пользователя.

"Jinn-Client" представляет собой сервис формирования электронной подписи и доверенной визуализации документа с помощью COM-технологии. Физически

представляет собой COM-сервер, размещенный вне процесса в COM-контейнере JinnService.exe.

При старте JinnService регистрируется в компоненте Windows SCM как COM-объект синглтон (один объект на всех клиентов). Для осуществления параллельного доступа нескольких пользователей к JinnService введено понятие криптографический контекст. Он используется для хранения состояния работы пользователя и синхронизации доступа к доверенной среде.

Криптографические контексты предоставляются пользователю в виде COM-интерфейсов. Определены несколько контекстов исходя из типа входных данных.

Для удобства интеграции "Jinn-Client" в современные информационные системы, основанные на Web, API реализован так, чтобы обратиться к нему можно было из любого браузера на языке JavaScript.

Основной рабочий интерфейс "Jinn-Client" — это IJinnService, который создает криптоконтексты, формирующие электронную подпись. В разделе "Методы и свойства объектаJinn-Client" приводится подробное описание используемых свойств и методов. Примеры скриптов использования COM-объекта для создания подписи XML-документа, CMS-подписи и подписи на основе хэша документа приводятся в файлах **XML.js**, **CMS.js** и **hashCMS.js** соответственно.

"Jinn-Client" обеспечивает визуальное представление подписываемого экземпляра формуляра с учетом XSLT-преобразования, реализацию криптографических функций хэширования, формирование ЭП. В соответствии с требованиями регуляторов в сфере информационной безопасности криптографические функции реализуются в соответствии с алгоритмами, описанными ГОСТ Р 34.10–2001, ГОСТ Р 34.11–94.

"Jinn-Client" работает в двух режимах подписи: в режиме принудительной визуализации и в режиме визуализации по требованию пользователя. Визуализация доступна для документов, преобразованных на основе XSL-файлов, переданных вместе с документом. Дополнительно к визуализированному документу пользователю предоставляется возможность просмотреть его наименование и сведения о владельце сертификата ключа проверки электронной подписи, с использованием которого осуществляется формирование ЭП. В окне визуализации пользователь может выполнять поиск информации и просмотр содержимого подписываемого документа. Предусмотрена возможность подписи любых форматов документов. Визуализация доступна только для документов типа MIMETYPE-TEXT.

В процессе создания ЭП пользователь может выбрать сертификат ключа проверки ЭП, который будет использоваться для подписания электронного документа, и просматривать информацию о сертификате.

В процессе создания ЭП документа пользователь с помощью компонента "Jinn-Client" просматривает содержание подписываемой информации (при этом в окне визуализации подписываемых данных компонент обеспечивает функции поиска и просмотра содержимого), подтверждает создание ЭП и получает уведомление в экранной форме об успешном результате завершения операции по созданию ЭП.

Модуль JBP

Для обеспечения связи между "Jinn-Client" и веб-браузером разработан специализированный программный модуль JBP. Модуль JBP устанавливается на рабочее место пользователя с помощью единой программы установки.

Работа с "Jinn-Client" (вызов документа, его отображение на экране, подписание, проверка ЭП) посредством JBP осуществляется пользователем с помощью веб-браузера (Internet Explorer/Mozilla Firefox/Google Chrome/Opera). При использовании браузера Internet Explorer, поддерживающего технологию ActiveX, в ПАК "Jinn" реализована возможность связи с ПО "Jinn-Client" напрямую, без применения модуля JBP. Задача создания скрипта для веб-браузера по ис-

пользованию модуля JBP возлагается на сотрудника, выполняющего роль программиста.

В разделе "Методы и свойства объекта плагина" приводится описание методов и свойств, используемых для создания скрипта для веб-браузера. Модуль JBP может возвращать ошибку двумя способами: через код возврата и через механизм исключений. Механизм исключений поддерживается только в браузере Internet Explorer.

В комплексе ПАК "Jinn" функции формирования ЭП документа, контрольного тестирования (проверки) ЭП, визуализации подписываемого документа реализуются с помощью программного компонента "Jinn-Client". Модуль JBP устанавливается в составе ПО "Jinn-Client".

Глава 2
Использование "Jinn-Client"

Настройка отображения заголовков в окне визуализации "Jinn-Client"

Синтаксис заголовка

[Title id='123' number='123' parentId='123' name='ABC']

Квадратные скобки и одинарные кавычки делают метку невидимой для XML-парсеров, что позволяет вставлять их внутрь значений XML-тэгов или их атрибутов.

"Jinn-Client" детектирует метку с помощью регулярного выражения \\[Title.*?\\], между [и Title не должно быть пробела.

Поиск полей внутри метки производится по следующим регулярным выражениям:

\\bid='\\z'

\\bnumber='\\z'

\\bparentId='\\z'

\\bname='.*'

Справа и слева от знака = не должно быть пробелов. Значения числовых полей (id, number, parentId) также не должны содержать пробелов.

Поля внутри метки допустимо переставлять местами.

Допустимые значения полей

Поля id должны содержать неповторяющиеся положительные числовые значения в пределах документа начиная с 1.

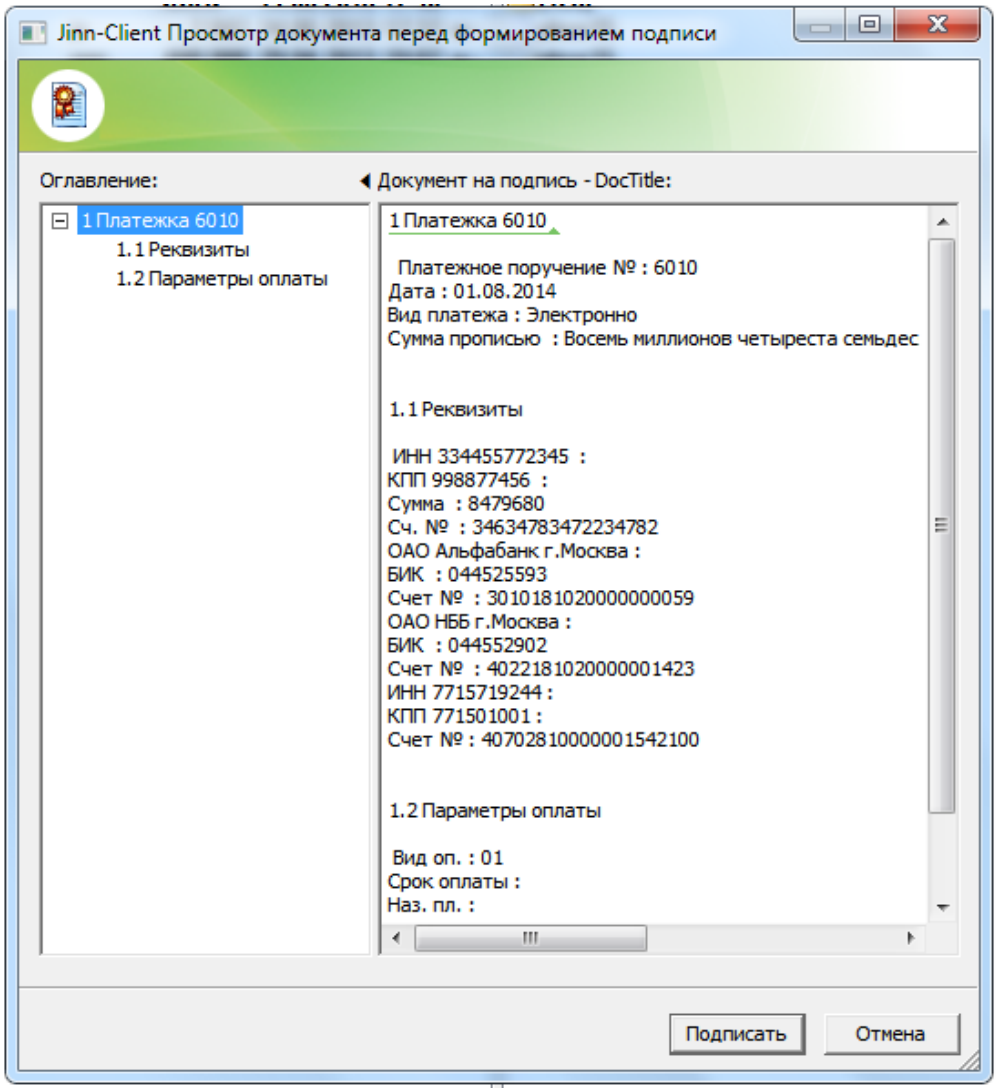
Поле parentId является ссылкой на существующее в документе значение поля id родительской метки. Если метка является корневой, то значение parentId должно быть 0.

Поле number — любое положительное число, задающее номер заголовка отображаемого в окне навигации оглавления. "Jinn-Client" рекурсивно проходит по иерархии заголовков, собирая поля number, и формирует составной номер заголовка типа "1.1.1".

Поле name может принимать произвольное строковое значение. "Jinn-Client" подставляет значение этого поля через пробел после составного номера заголовка, таким образом на экране отображается название типа "1.1.1 Реквизиты".

Ниже представлены пример XML-документа и соответствующее ему окно:

```
<?xml version="1.0" encoding="utf-8"?>
<doc>
  <content id="SecurityCode">
    <field sign="1" order="0" label=" [Title id='1' number='1' parentId='0' name='Платежка 6010'] Платежное поручение №" value="6010"/>
    <field sign="1" order="1" label="Дата" value="01.08.2012"/>
    <field sign="1" order="2" label="Вид платежа" value="Электронно"/>
    <field sign="1" order="3" label="Сумма прописью " value="Восемь миллионов четыреста семьдесят девять тысяч шестьсот восемьдесят рублей 00 копеек "/>
    <field sign="1" order="4" label=" [Title id='2' number='1' parentId='1' name='Реквизиты'] ИНН 334455772345 "/>
    <field sign="1" order="5" label="КПП 998877456 "/>
    <field sign="1" order="6" label="Сумма " value="8479680"/>
    <field sign="1" order="7" label="Сч. № " value="34634783472234782"/>
    <field sign="1" order="8" label="ОАО Альфабанк г.Москва"/>
    <field sign="1" order="9" label="БИК " value="044525593 "/>
    <field sign="1" order="10" label="Счет № " value="30101810200000000059 "/>
    <field sign="1" order="11" label="ОАО НББ г.Москва"/>
    <field sign="1" order="12" label="БИК " value="044552902 "/>
    <field sign="1" order="13" label="Счет № " value="40221810200000001423 "/>
    <field sign="1" order="14" label="ИНН 7715719244"/>
    <field sign="1" order="15" label="КПП 771501001"/>
    <field sign="1" order="16" label="Счет №" value="40702810000001542100"/>
    <field sign="1" order="17" label=" [Title id='3' number='2' parentId='1' name='Параметры оплаты'] Вид оп." value="01"/>
    <field sign="1" order="18" label="Срок оплаты" value=" "/>
    <field sign="1" order="19" label="Наз. пл." value=" "/>
    <field sign="1" order="20" label="Очер. оплаты" value="6"/>
    <field sign="1" order="21" label="Код" value=" "/>
    <field sign="1" order="22" label="Рез. поле" value=" "/>
  </content>
</doc>
```

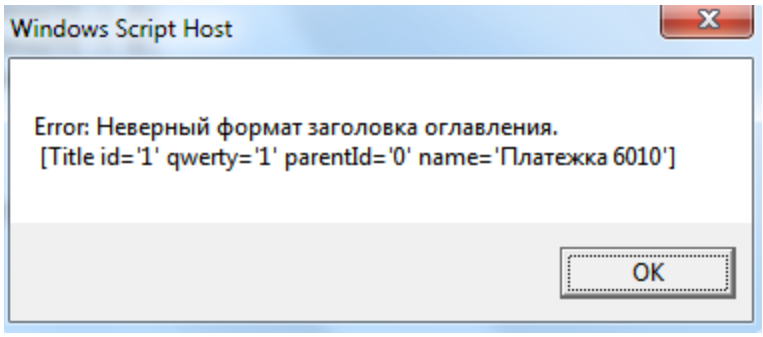


Проверка программы

Синтаксические ошибки

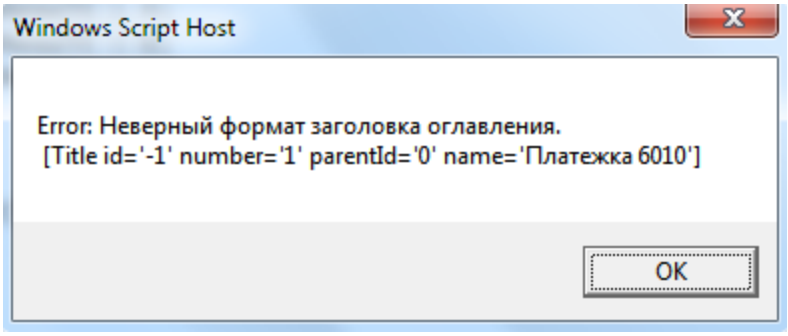
При наличии синтаксической ошибки в открывающем ([Title) или закрывающем (]) элементе метки "Jinn-Client" пропустит наличие заголовка. Если на такой заголовок ссылаются другие (дочерние) заголовки, то его пропуск приводит к структурной ошибке.

Наличие синтаксической ошибки в полях метки равносильно отсутствию такого поля. Процедура формирования подписи завершится ошибкой следующего вида:



Ошибки допустимых значений

При недопустимом значении поля метки процедура формирования подписи завершится ошибкой следующего вида:



Структурные ошибки

1. При наличии дочерних заголовков корневому элементу назначен parentId > 0.

   [Title id='1' number='1' parentId='23' name='Платежка 6010']

   [Title id='2' number='1' parentId='1' name='Реквизиты']

   [Title id='3' number='2' parentId='1' name='Параметры оплаты']

2. Дочерний заголовок ссылается на несуществующий id.

   [Title id='1' number='1' parentId='0' name='Платежка 6010']

   [Title id='2' number='1' parentId='23' name='Реквизиты']

   [Title id='3' number='2' parentId='1' name='Параметры оплаты']

В перечисленных случаях процедура формирования подписи завершится сообщением об ошибке.

Методы и свойства объекта "Jinn-Client"

Методы интерфейса IJinnService:

- метод **CreateXmlCryptoContext()** — создает криптоконтекст для подписи XML-документов;
- метод **CreateBinaryBase64CryptoContext()** — создает криптоконтекст для подписи двоичного и текстового документа.

Метод CreateXmlCryptoContext

XML-криптоконтекст имеет следующие функции:

- **SetProperty (property_ name, property_ value)** — задает свойства объекта: указывается имя (property_ name) и значение (property_ value) свойства. Ниже в таблице представлен перечень возможных используемых свойств:

property_name	property_value
JinnOwnerCertificateName — имя (ФИО) владельца сертификата	Строка
JinnOwnerCertificatePosition — должность владельца сертификата	Строка
JinnTitleDocument – заголовок документа	Строка
ShowOperationResult – отображать/ не отображать сообщения "Jinn-Client"	Сообщения не отображаются, если свойство имеет значение false

- **GetProperty (property_ name)** — возвращает значение свойства (см. таблицу выше).
- **SignDocumentWithTransform (document, element_id, xslt, xslt_URI, signed_properties, unsigned_properties, namespace_ref, namespace_ prefix, visualization)** — возвращает подпись документа в формате XsadesBES, зашифрованную посредством алгоритма Base64.

  Параметры функции **SignDocumentWithTransform**:

- document — подписываемый документ, зашифрованный Base64;
- element_id — значение атрибута id узла подписываемого документа. Если данный параметр пуст, то будет подписан весь документ;
- xslt — xslt-схема преобразования документа при формировании подписи;
- xslt_URI — URI xslt-схемы. Если данный параметр пуст, то в подпись будет вставлена вся схема целиком;
- signed_properties — подписываемые пользовательские атрибуты;
- unsigned_properties — неподписываемые пользовательские атрибуты;
- namespace_ref — ссылка на пространство имен для пользовательских атрибутов;
- namespace_prefix — префикс пространства имен для пользовательских атрибутов;
- visualization — визуализация документа (true — визуализировать, false — не визуализировать).

- **CheckSignature(document, xslt, signature, certificate, visualization)** — проверяет правильность подписи. Если подпись верна, возвращает true, если подпись неверна — false.

  Параметры функции **CheckSignature**:

  - document — подписанный документ, зашифрованный Base64;
  - xslt — xslt-схема преобразования документа при формировании подписи;
  - signature — подпись документа document, зашифрованная Base64;
  - certificate — зашифрованный Base64 сертификат, содержащий открытый ключ;
  - visualization — визуализация документа (true — визуализировать, false — не визуализировать).

Метод CreateBinaryBase64CryptoContext

BinaryBase64-криптоконтекст имеет следующие функции:

- **SetProperty (property_ name, property_ value)** — задает свойства объекта: указывается имя (property_ name) и значение (property_ value) свойства. Ниже в таблице представлен перечень возможных используемых свойств:

property_name	property_value
JinnOwnerCertificateName — имя (ФИО) владельца сертификата	Строка
JinnOwnerCertificatePosition — должность владельца сертификата	Строка
JinnTitleDocument — заголовок документа	Строка
JinnIncludeCertificateToCMS — размещение сертификата в CMS-контейнере	Сертификат помещается в контейнер, если передается строка true
JinnIncludeDocumentToCMS — размещение документа в CMS-контейнере	Документ помещается в контейнер, если передается строка true
ShowOperationResult — отображать/ не отображать сообщения "Jinn-Client"	Сообщения не отображаются, если свойство имеет значение false

- **GetProperty (property_ name)** — возвращает значение свойства (см. таблицу выше).
- **SignDocument (document, MIME_ string, attributes_ to_ sign, attributes_not_sign, visualization)** — возвращает зашифрованный Base64 CMS-контейнер с подписью документа document.

  Параметры функции **SignDocument**:

  - document — подписываемый документ, зашифрованный Base64;
  - MIME_string — MIME-строка, описывающая тип документа;

> **Примечание.** Если MIME_ string содержит последовательность символов "text", то подписываемые данные считаются текстом. Текст будет визуализироваться, если выставлен флаг визуализации.
>
> В противном случае подписываемые данные считаются двоичными и визуализироваться не будут. Пользователь увидит сообщение о невозможности визуализации подписываемых двоичных данных.

- attributes_to_sign — подписываемые атрибуты;
- attributes_not_sign — неподписываемые атрибуты;
- visualization — визуализация документа (true — визуализировать, false — не визуализировать).

- **SignHashDocument (hash_ document, MIME_ string, attributes_ to_ sign, attributes_not_sign)** — возвращает зашифрованный Base64 CMS-контейнер с подписью хэш-кода документа document.

  Параметры функции **SignHashDocument**:

  - hash_document — подписываемый хэш-код документа, зашифрованный Base64;
  - MIME_ string — MIME- строка, описывающая тип документа (см. примечание выше);
  - attributes_to_sign — подписываемые атрибуты;
  - attributes_not_sign — неподписываемые атрибуты.
- **CheckSignature(document, MIME_string, CMS_container, certificate, visualization)** — проверяет правильность подписи. Если подпись верна, возвращает true, если подпись неверна — false.

  Параметры функции **CheckSignature**:

  - document — подписанный документ, зашифрованный Base64;
  - MIME_ string — MIME- строка, описывающая тип документа (см. примечание выше);
  - CMS_container — зашифрованный Base64 контейнер с подписью;
  - certificate — зашифрованный Base64 сертификат, содержащий открытый ключ;
  - visualization — визуализация документа (true — визуализировать, false — не визуализировать).
- **CheckHashSignature(Hash_document, signature, certificate)** — проверяет правильность подписи хэш- кода документа. Если подпись верна, возвращает true, если неверна — false.

  Параметры функции **CheckHashSignature**:

  - hash_document — подписываемый хэш-код документа, зашифрованный Base64;
  - MIME_ string — MIME- строка, описывающая тип документа (см. примечание выше);
  - CMS_container — зашифрованный Base64 контейнер с подписью;
  - certificate — зашифрованный Base64 сертификат, содержащий открытый ключ.

Пример реализации объекта "Jinn-Client"

Приведенный ниже листинг содержит пример реализации функции выработки отсоединенной ЭП в формате XAdES-BES с использованием "Jinn-Client" на языке JavaScript.

```
function sign_ document (xmlText, xmlID, xslt, xsltURI, signed_ properties, unsigned_properties, namespace_reF, namespase_preFix, needVisualize)
{
   try
   {
      // Создаём объект, через который доступно всё API Jinn-Client'a.
      var JinnService = new ActiveXObject("JinnServiceLibrary.JinnService");
      // Получаем контекст для работы с XML-документами.
      var JinnXmlCryptoContext = JinnService.CreateXmlCryptoContext();

      // Подписываем XML файл.
      var Signature = JinnXmlCryptoContext.SignDocumentWithTransform(xmlText, xmlID, xslt, xsltURI, signed_ properties, unsigned_ properties, namespace_ reF, namespase_preFix, needVisualize);

      alert("Документ подписан, подпись:\r\n" + Signature);
      // Проверяем подпись (алгоритмическая проверка без проверки серти-фиката)
      // Считаем, что сертификат заранее был загружен в переменную Base64Certificate.
      var bResult = JinnXmlCryptoContext.CheckSignature(xmlText, xmlID, xslt, Signature, Base64Certificate, needVisualize);

      if( bResult )
      {
alert("Подпись проверена!");
      }
      else
      {
         alert("Подпись не прошла проверку!");
      }
      JinnXmlCryptoContext.Close();
   }
   catch(ex)
   {
      alert(ex.name + ": " + ex.message);
   }
}
```

Методы и свойства объекта плагина

Объект модуля JBP (далее — объект плагина) создается вызовом метода **document.getElementByID ("id")**. В процессе разработки используются:

- метод **CreateXmlCryptoContext()** — создает криптоконтекст для подписи XML-документов;
- метод **CreateXmlCryptoContextAsync()** — создает криптоконтекст для асинхронной подписи XML-документов;
- метод **CreateBinaryBase64CryptoContext()** — создает криптоконтекст для подписи двоичного и текстового документа;
- метод **CreateBinaryBase64CryptoContextAsync()** — создает криптоконтекст для подписи двоичного и текстового документа;
- свойство **LastError** — используется для получения описания и кода ошибки при использовании синхронного API;
- свойство **ThrowException** — если свойству присвоено булевское значение true, то при возникновении ошибки будет отбрасываться исключение при использовании синхронного API.

Метод CreateXmlCryptoContext

XML-криптоконтекст имеет следующие функции:

- **SetJinnProperty(property_name, property_value)** — задает свойства объекта: указывается имя (property_name) и значение (property_value) свойства. Ниже в таблице представлен перечень возможных используемых свойств.

property_name	property_value
JinnOwnerCertificateName — имя (ФИО) владельца сертификата	Строка
JinnOwnerCertificatePosition — должность владельца сертификата	Строка
JinnTitleDocument — заголовок документа	Строка
ShowOperationResult — отображать/ не отображать сообщения "Jinn-Client"	Сообщения не отображаются, если свойство имеет значение false

- **GetJinnProperty(property_name)** — возвращает значение свойства (см. таблицу выше).
- **SignDocumentWithTransform(document, element_id, xslt, xslt_URI, signed_properties, unsigned_properties, namespace_ref, namespace_prefix, visualization)** — возвращает подпись документа в формате XsadesBES, зашифрованную посредством алгоритма Base64.

  Параметры функции **SignDocumentWithTransform**:

  - document — подписываемый документ, зашифрованный Base64;
  - element_id — значение атрибута id узла подписываемого документа. Если данный параметр пуст, то будет подписан весь документ;
  - xslt — xslt-схема преобразования документа при формировании подписи;
  - xslt_URI — URI xslt-схемы. Если данный параметр пуст, то в подпись будет вставлена вся схема целиком;
  - signed_properties — подписываемые пользовательские атрибуты;
  - unsigned_properties — неподписываемые пользовательские атрибуты;
  - namespace_ref — ссылка на пространство имен для пользовательских атрибутов;
  - namespace_prefix — префикс пространства имен для пользовательских атрибутов;

- visualization — визуализация документа (true — визуализировать, false — не визуализировать).

- **CheckSignature(document, xslt, signature, certificate, visualization)** — проверяет правильность подписи. Если подпись верна, возвращает true, если подпись неверна — false.

  Параметры функции **CheckSignature**:

  - document — подписанный документ, зашифрованный Base64;
  - xslt — xslt-схема преобразования документа при формировании подписи;
  - signature — подпись документа document, зашифрованная Base64;
  - certificate — зашифрованный Base64 сертификат, содержащий открытый ключ;
  - visualization — визуализация документа (true — визуализировать, false — не визуализировать).

Метод CreateBinaryBase64CryptoContext

BinaryBase64-криптоконтекст имеет следующие функции:

- **SetJinnProperty(property_name, property_value)** — задает свойства объекта: указывается имя (property_name) и значение (property_value) свойства. Ниже в таблице представлен перечень возможных используемых свойств.

property_name	property_value
JinnOwnerCertificateName — имя (ФИО) владельца сертификата	Строка
JinnOwnerCertificatePosition — должность владельца сертификата	Строка
JinnTitleDocument — заголовок документа	Строка
JinnIncludeCertificateToCMS — размещение сертификата в CMS-контейнере	Сертификат помещается в контейнер, если передается строка true
JinnIncludeDocumentToCMS — размещение документа в CMS-контейнере	Документ помещается в контейнер, если передается строка true
ShowOperationResult — отображать/ не отображать сообщения "Jinn-Client"	Сообщения не отображаются, если свойство имеет значение false

- **GetJinnProperty(property_name)** — возвращает значение свойства (см. таблицу выше).

- **SignDocument (document, MIME_ string, attributes_ to_ sign, attributes_not_sign, visualization)** — возвращает зашифрованный Base64 CMS-контейнер с подписью документа document.

  Параметры функции **SignDocument**:

  - document — подписываемый документ, зашифрованный Base64;
  - MIME_string — MIME-строка, описывающая тип документа;

    Примечание. Если MIME_ string содержит последовательность символов "text", то подписываемые данные считаются текстом. Текст будет визуализироваться, если выставлен флаг визуализации.
    В противном случае подписываемые данные считаются двоичными и визуализироваться не будут. Пользователь увидит сообщение о невозможности визуализации подписываемых двоичных данных.

  - attributes_to_sign — подписываемые атрибуты;
  - attributes_not_sign — неподписываемые атрибуты;
  - visualization — визуализация документа (true — визуализировать, false — не визуализировать).

- **SignHashDocument (hash_ document, MIME_ string, attributes_ to_ sign, attributes_not_sign)** — возвращает зашифрованный Base64 CMS-контейнер с подписью хэш-кода документа document.

  Параметры функции **SignHashDocument**:

  - hash_document — подписываемый хэш-код документа, зашифрованный Base64;
  - MIME_ string — MIME- строка, описывающая тип документа (см. примечание выше);
  - attributes_to_sign — подписываемые атрибуты;
  - attributes_not_sign — неподписываемые атрибуты.
- **CheckSignature(document, MIME_string, CMS_container, certificate, visualization)** — проверяет правильность подписи. Если подпись верна, возвращает true, если подпись неверна — false.

  Параметры функции **CheckSignature**:

  - document — подписанный документ, зашифрованный Base64;
  - MIME_ string — MIME- строка, описывающая тип документа (см. примечание выше);
  - CMS_container — зашифрованный Base64 контейнер с подписью;
  - certificate — зашифрованный Base64 сертификат, содержащий открытый ключ;
  - visualization — визуализация документа (true — визуализировать, false — не визуализировать).
- **CheckHashSignature(Hash_document, signature, certificate)** — проверяет правильность подписи хэш- кода документа. Если подпись верна, возвращает true, если неверна — false.

  Параметры функции **CheckHashSignature**:

  - hash_document — подписываемый хэш-код документа, зашифрованный Base64;
  - MIME_ string — MIME- строка, описывающая тип документа (см. примечание выше);
  - CMS_container — зашифрованный Base64 контейнер с подписью;
  - certificate — зашифрованный Base64 сертификат, содержащий открытый ключ.

Метод CreateXmlCryptoContextAsync

Асинхронный XML-криптоконтекст имеет следующие функции:

- **SignDocumentWithTransform(data, successCallback, errorCallback)** — инициирует подпись документа в формате XsadesBES, зашифрованную посредством алгоритма Base64.

  Параметры функции **SignDocumentWithTransform**:

  - data — JSON- строка, содержащая JS- объект с ключами и соответствующими значениями:
    - doc — подписываемый документ, зашифрованный Base64;
    - xslt — xslt- схема преобразования документа при формировании подписи;
    - options — JS-объект с опциональными параметрами.

    Описание допустимых ключей объекта **options**:

    - elementUri — значение атрибута id узла подписываемого документа. Если данный параметр пуст, то будет подписан весь документ (строка в UTF-8);
    - xsltUri — URI xslt-схемы. Если данный параметр пуст, то в подпись будет вставлена вся схема целиком (строка в UTF-8);

- signedProperties — подписываемые пользовательские атрибуты (строка в UTF-8 в формате "name1:value1&name2:value2&name3:value3");
- unsignedProperties — неподписываемые пользовательские атрибуты (строка в UTF-8 в формате "name1:value1&name2:value2&name3:value3");
- propertiesNamespaceRef — ссылка на пространство имен для пользовательских атрибутов (строка в UTF-8);
- propertiesNamespacePrefix — префикс пространства имен для пользовательских атрибутов (строка в UTF-8);
- needVisualize — визуализация документа (true — визуализировать, false — не визуализировать);
- showResult — отображение сообщений "Jinn- Client" (true — отображать, false — не отображать);
- documentTitle — заголовок документа (строка в UTF-8);
- ownerCertificateName — имя (ФИО) владельца сертификата (строка в UTF-8);
- ownerCertificatePosition — должность владельца сертификата (строка в UTF-8);
- certificate — зашифрованный Base64 сертификат, содержащий открытый ключ (при наличии параметра выполняется проверка подписи);

- successCallback — функция, которая будет вызвана при успешной подписи документа. В качестве параметра получает JS-объект с ключами:
  - result — равен 0 при успешной подписи;
  - signature — строка с подписью, зашифрованной с помощью алгоритма Base64;
- errorCallback — функция, которая будет вызвана в случае ошибки. В качестве параметра получает JS-объект с ключами:
  - result — код ошибки;
  - signature — строка с подписью, зашифрованной с помощью алгоритма Base64 (присутствует, если подпись была сформирована успешно, однако проверка подписи завершилась с ошибкой);
  - error — описание ошибки.

Метод CreateBinaryBase64CryptoContextAsync

Асинхронный BinaryBase64-криптоконтекст имеет следующие функции:

- **SignDocument (data, successCallback, errorCallback)** — инициирует подпись документа, зашифрованного посредством алгоритма Base64.

  Параметры функции **SignDocument**:

  - data — JSON- строка, содержащая JS- объект с ключами и соответствующими значениями:
    - doc — подписываемый документ, зашифрованный Base64;
    - options — JS-объект с опциональными параметрами.

  Описание допустимых ключей объекта **options**:

  - mimeType — MIME-строка, описывающая тип документа (строка в UTF-8);
  - signedAttributes — подписываемые пользовательские атрибуты (строка в UTF-8 в формате "name1:value1&name2:value2&name3:value3");
  - unsignedAttributes — неподписываемые пользовательские атрибуты (строка в UTF-8 в формате "name1:value1&name2:value2&name3:value3");

- needVisualize — визуализация документа (true — визуализировать, false — не визуализировать);
- showResult — отображение сообщений "Jinn- Client" (true — отображать, false — не отображать);
- documentTitle — заголовок документа (строка в UTF-8);
- ownerCertificateName — имя (ФИО) владельца сертификата (строка в UTF-8);
- ownerCertificatePosition — должность владельца сертификата (строка в UTF-8);
- includeCertificateToCMS — размещение сертификата в CMS-контейнере (true — помещать, false — не помещать);
- includeDocumentToCMS — размещение документа в CMS-контейнере (true — помещать, false — не помещать);
- certificate — зашифрованный Base64 сертификат, содержащий открытый ключ (при наличии параметра выполняется проверка подписи);

- successCallback — функция, которая будет вызвана при успешной подписи документа. В качестве параметра получает JS-объект с ключами:
  - result — равен 0 при успешной подписи;
  - signature — CMS-контейнер с подписью документа, зашифрованный с помощью алгоритма Base64;
- errorCallback — функция, которая будет вызвана в случае ошибки. В качестве параметра получает JS-объект с ключами:
  - result — код ошибки;
  - signature — CMS-контейнер с подписью документа, зашифрованный с помощью алгоритма Base64 (присутствует, если подпись была сформирована успешно, однако проверка подписи завершилась с ошибкой);
  - error — описание ошибки;

- **SignHashDocument (data, successCallback, errorCallback)** — инициирует подпись хэш-кода документа.

  Параметры функции **SignHashDocument**:

  - data — JSON- строка, содержащая JS- объект с ключами и соответствующими значениями:
    - hash — подписываемый документ, зашифрованный Base64;
    - options — JS-объект с опциональными параметрами.

  Описание допустимых ключей объекта **options**:

  - mimeType — MIME-строка, описывающая тип документа (строка в UTF-8);
  - signedAttributes — подписываемые пользовательские атрибуты (строка в UTF-8 в формате "name1:value1&name2:value2&name3:value3");
  - unsignedAttributes — неподписываемые пользовательские атрибуты (строка в UTF-8 в формате "name1:value1&name2:value2&name3:value3");
  - showResult — отображение сообщений "Jinn- Client" (true — отображать, false — не отображать);
  - documentTitle — заголовок документа (строка в UTF-8);
  - ownerCertificateName — имя (ФИО) владельца сертификата (строка в UTF-8);
  - ownerCertificatePosition — должность владельца сертификата (строка в UTF-8);

- includeCertificateToCMS — размещение сертификата в CMS-контейнере (true — помещать, false — не помещать);
- certificate — зашифрованный Base64 сертификат, содержащий открытый ключ (при наличии параметра выполняется проверка подписи);

- successCallback — функция, которая будет вызвана при успешной подписи документа. В качестве параметра получает JS-объект с ключами:
  - result — равен 0 при успешной подписи;
  - signature — CMS-контейнер с подписью хэш-кода документа, зашифрованный с помощью алгоритма Base64;
- errorCallback — функция, которая будет вызвана в случае ошибки. В качестве параметра получает JS-объект с ключами:
  - result — код ошибки;
  - signature — CMS-контейнер с подписью хэш-кода документа, зашифрованный с помощью алгоритма Base64 (присутствует, если подпись была сформирована успешно, однако проверка подписи завершилась с ошибкой);
  - error — описание ошибки.

Свойство LastError

XML-криптоконтекст и BinaryBase64-криптоконтекст имеют свойство **LastError**, с помощью которого можно получить код ошибки и ее описание. Доступ осуществляется посредством свойств ErrorCode и ErrorMessage:

- свойство ErrorCode — возвращает код ошибки, имеет ненулевое значение в случае наличия ошибки;
- свойство ErrorMessage — возвращает описание ошибки.

Пример реализации объекта плагина

Инициализация

Объявление компонента:

```
...
<object id="jinn" type="application/x-vnd.JinnBrowserPlugin" width="0"
height="0">
   <param name="ThrowException" value="true" />
</object>
...
```

С помощью параметра ThrowException задается стратегия обработки ошибок. Если value выставлено в true, то компонент будет генерировать исключения в случае ошибок. Если value выставлено в false, то код и описания ошибки будут содержаться в свойстве LastError.

Создание экземпляра компонента:

```
...
function $(name) {
   return document.getElementById(name);
}
var jinn;
function load() {
   jinn = $('jinn');
}
...
```

Load обработчик загрузки документа.

```
...
<BODY class="qqq" onload="load()">
...
```

Использование плагина с анализом кода ошибки

Создание криптоконтекстов и подписывание документа:

```
function sign_document_plugin() {
   var JinnXmlCryptoContext = jinn.CreateXmlCryptoContext();
   if (jinn.LastError.ErrorCode != 0) {
      window.alert(jinn.LastError.ErrorMessage);
      return;
   }
   JinnXmlCryptoContext.SetJinnProperty("JinnTitleDocument", "DocTitle");
   // подпись документа с трансформацией
   // подписывает часть документа, передаваемая xslt схем рассчитана на документ, вынутый из передаваемого по id (id = SecurityCode) узла.
   var Signature = JinnXmlCryptoCon- text.SignDocumentWithTransform (base64UTF8Document, "SecurityCode", base64UTF8Xslt, "XSLT URI", signed_properties, unsigned_properties, namespace_reF, namespase_preFix, true);
   if (JinnXmlCryptoContext.LastError.ErrorCode != 0) {
      window.alert(JinnXmlCryptoContext.LastError.ErrorMessage);
      JinnXmlCryptoContext.Close();
      return;
   }
   // проверка подписи
   var bResult = JinnXmlCryptoContext.CheckSignature (base64UTF8Document, base64UTF8Xslt, Signature, base64Certificate, true);
   if (JinnXmlCryptoContext.LastError.ErrorCode != 0)
   {
      window.alert(JinnXmlCryptoContext.LastError.ErrorMessage);
      JinnXmlCryptoContext.Close();
      return;
   }
   if (bResult) {
      alert("Подпись проверена!");
   }
   else {
      alert("Подпись не прошла проверку!");
   }
   JinnXmlCryptoContext.Close();
}
```

Плагин JinnSignExtensionProvider

Взаимодействие с плагином происходит посредством вызова функций из скрипта **JinnSignExtensionProvider.js.** Скрипт содержит описание класса JinnSignExtensionProvider, который предоставляет функции для создания криптоконтекстов для подписи документов:

- **CreateXmlCryptoContext = function ()** — возвращает объект класса CryptoContextXML для подписи XML-документов.
- **CreateBinaryBase64CryptoContext = function ()** — возвращает объект класса CryptoContextBinaryBase64 для CMS-подписи документов или хэш-кода документов.

- **CheckExtension = function()** — возвращает значение объекта функции Js-promise. При вызове происходит проверка поддержки браузером расширения NPAPI или PPAPI и, в случае наличия плагина соответствующего типа, promise успешно выполняется. При отсутствии необходимого расширения promise отклоняется с ответом в виде объекта с полями error и type. Error содержит описание ошибки, type – тип плагина PPAPI или NPAPI.

Описание класса CryptoContextXML

Класс CryptoContextXML предоставляет асинхронные функции для работы с XML криптоконтекстом:

- **SignDocumentWithTransform = function(document, xslt, options)** — подпись одного экземпляра документа.

Параметр	Тип	Описание
Параметры функции SignDocumentWithTransform		
document	Строковый (base64)	Документ для подписи
xslt	Строковый (base64)	Шаблон для преобразования XML-документа
options		Объект с необязательными опциональными параметрами (см. ниже)
Опциональные параметры функции SignDocumentWithTransform		
showResult	Булевый (true/false)	Отображение результатов операции (по умолчанию true — включено)
needVisualize	Булевый (true/false)	Отображение подписываемого документа (по умолчанию true — включено)
elementUri	Строковый (UTF-8)	ID узла XML (если указан — для подписания преобразуется часть документа из соответствующего узла, если не задан — подписывается весь документ)
xsltUri	Строковый (UTF-8)	Адрес XSLT-шаблона, который указывается в преобразованном документе (если не задан — в итоговый XML добавляется весь XSLT-шаблон)
signedProperties	Строковый (UTF-8)	Подписываемые атрибуты в формате "name1:value1&name2:value2&name3:value3" (по умолчанию не заданы)
unsignedProperties	Строковый (UTF-8)	Неподписываемые атрибуты в формате "name1:value1&name2:value2&name3:value3" (по умолчанию не заданы)
propertiesNamespaceRef	Строковый (UTF-8)	Пространство имен, добавляемое к параметру signedProperties (по умолчанию не задано)
propertiesNamespacePrefix	Строковый (UTF-8)	Префикс пространства имен, добавляемого к параметру signedProperties (по умолчанию не задан)
documentTitle	Строковый (UTF-8)	Заголовок, который будет показан при отображении документа (по умолчанию не задан)
ownerCertificateName	Строковый (UTF-8)	Имя пользователя, которое будет показано при отображении документа (по умолчанию не задано)
ownerCertificatePosition	Строковый (UTF-8)	Должность пользователя, которая будет показана при отображении документа (по умолчанию не задана)
certificate	Строковый (base64)	Сертификат для проверки подписи (при его наличии после подписи документа выполняется тестовая проверка подписи, при отсутствии параметра — проверка не выполняется)

- **Close = function()** — завершить работу с криптоконтекстом (после вызова функции все новые вызовы будут отклоняться, все незавершенные вызовы, ожидающие своей очереди, будут обработаны).

Ответом функции SignDocumentWithTransform выступает объект функции Js-promise, который в качестве параметра получает JS-объект с ключами:

При успешной подписи документа		**В случае ошибки**	
Ключ	**Значение**	**Ключ**	**Значение**
result	0	result	Код ошибки (!= 0)
signature	Строка с подписью в формате base64	error	Строка с описанием ошибки в формате UTF-8

Описание класса CryptoContextBinaryBase64

Класс CryptoContextBinaryBase64 предоставляет асинхронные функции для работы с BinaryBase64 криптоконтекстом:

- **SignDocument = function (document, options)** — подпись одного экземпляра документа.

Параметр	**Тип**	**Описание**
Параметры функции SignDocument		
document	Строковый (base64)	Документ для подписи
options		Объект с необязательными опциональными параметрами (см. ниже)
Опциональные параметры функции SignDocument		
showResult	Булевый (true/false)	Отображение результатов операции (по умолчанию true — включено)
needVisualize	Булевый (true/false)	Отображение подписываемого документа (по умолчанию true — включено)
mimeType	Строковый (UTF-8)	MIME-тип документа: программа, аудиофайл, изображение, текст или видеофайл (по умолчанию text)
signedAttributes	Строковый (UTF-8)	Подписываемые атрибуты в формате "name1:value1&name2:value2&name3:value3" (по умолчанию не заданы)
unsignedAttributes	Строковый (UTF-8)	Неподписываемые атрибуты в формате "name1:value1&name2:value2&name3:value3" (по умолчанию не заданы)
documentTitle	Строковый (UTF-8)	Заголовок, который будет показан при отображении документа (по умолчанию не задан)
ownerCertificateName	Строковый (UTF-8)	Имя пользователя, которое будет показано при отображении документа (по умолчанию не задано)
ownerCertificatePosition	Строковый (UTF-8)	Должность пользователя, которая будет показана при отображении документа (по умолчанию не задана)
certificate	Строковый (base64)	Сертификат для проверки подписи (при его наличии после подписи документа выполняется тестовая проверка подписи, при отсутствии параметра — проверка не выполняется)

- **SignHashDocument = function(hash, options)** — подпись одного хэш-кода документа.

Параметр	Тип	Описание
Параметры функции SignHashDocument		
hash	Строковый (base64)	Хэш-код документа для подписи
options		Объект с необязательными опциональными параметрами (см. ниже)
Опциональные параметры функции SignHashDocument		
showResult	Булевый (true/false)	Отображение результатов операции (по умолчанию true — включено)
mimeType	Строковый (UTF-8)	MIME-тип документа: программа, аудиофайл, изображение, текст или видеофайл (по умолчанию text)
signedAttributes	Строковый (UTF-8)	Подписываемые атрибуты в формате "name1:value1&name2:value2&name3:value3" (по умолчанию не заданы)
unsignedAttributes	Строковый (UTF-8)	Неподписываемые атрибуты в формате "name1:value1&name2:value2&name3:value3" (по умолчанию не заданы)
certificate	Строковый (base64)	Сертификат для проверки подписи (при его наличии после подписи документа выполняется тестовая проверка подписи, при отсутствии параметра — проверка не выполняется)

- **Close = function()** — завершить работу с криптоконтекстом (после вызова функции все новые вызовы будут отклоняться, все незавершенные вызовы, ожидающие своей очереди, будут обработаны).

Ответом функции SignDocument или SignHashDocument выступает объект функции Js-promise, который в качестве параметра получает JS-объект с ключами:

При успешной подписи документа		В случае ошибки	
Ключ	**Значение**	**Ключ**	**Значение**
result	0	result	Код ошибки (!= 0)
signature	Строка с подписью в формате base64	error	Строка с описанием ошибки в формате UTF-8

Пример взаимодействия

Файл test.html:

```
<!DOCTYPE html>
<html lang="ru">
<head>
<meta charset="utf-8">
<meta name="viewport" content="width=device-width">
<title>Jinn Sign Extension Test</title>
<script src="jquery-1.11.3.min.js" type="text/javascript" charset="utf-
8"></script>
<script src="JinnSignExtensionProvider.js" type="text/javascript" charset="utf-
8"></script>
<script src="test.js" type="text/javascript" charset="utf-8"></script>
</head>
<body>
</body>
</html>
```

Файл test.js:

```
var base64doc = 'MQ==';
var options = {
needVisualize: false,
};
var jinn = new JinnSignExtensionProvider();
var cryptoContextBinaryBase64 = jinn.createBinaryBase64CryptoContext();
for (var i = 0; i < 100; i++) {
cryptoContextBinaryBase64.SignDocument (base64doc, options).then (function
(response) {
console.log(i + ' request signature: ' + response.signature);
}, function(response) {
console.error(i + ' request error:' + response.error);
});
}
cryptoContextBinaryBase64.Close();
```

Глава 3
Сообщения ПАК "Jinn"

Модуль JinnService.exe представляет собой COM-сервер, поэтому клиентский код должен уметь обнаруживать исключения типа _com_error и далее запрашивать у объекта-исключения объект ErrorInfo, с помощью которого можно получить код и строковое описание ошибки.

Модуль JinnService.exe генерирует исключения не только в случае ошибок в его бизнес-логике, но и в случае системных ошибок. В обоих вариантах свойство message объекта ErrorInfo будет содержать строковое описание ошибки. FACILITY CODE системных ошибок будет иметь значение FACILITY_NULL или FACILITY_WIN32, в случае ошибок в бизнес-логике значение будет 0x650. Подробнее об этом см. на сайте http://msdn.microsoft.com/ru-RU/library/windows/desktop/ms690088(v=vs.85).aspx.

Сообщения и ошибки уровня бизнес-логики

Ниже в таблице приведены коды ошибок уровня бизнес-логики и их описание:

Код ошибки	Символьное представление в C++	Описание
Информационные сообщения, отправляемые в EventLog системы Windows		
0x465003EA В EventLog MessageID = 1002	I_JINNSERVICE_ LOCALSERVER_ STARTED	"Jinn-Client" начал работу
0x065003EC В EventLog MessageID = 1004	S_JINNSERVICE_SERVER_ REGISTERED	"Jinn-Client" зарегистрирован
0x065003ED В EventLog MessageId = 1005	S_JINNSERVICE_ UNREGISTERED	Регистрация "Jinn-Client" удалена
Регистрация и дерегистрация COM-сервера, сообщения попадают в EventLog		
0xC65007D0 В EventLog MessageId=2000	E_JINNSERVICE_SERVICE_ HANDLER_NOT_ INSTALLED	Ошибка регистрации "Jinn-Client"
0xC65007D2 В EventLog MessageId=2002	E_JINNSERVICE_ REGISTER_LOCALSERVER_ FAILED	Ошибка регистрации "Jinn-Client"
0xC65007D3 В EventLog MessageId=2003	E_JINNSERVICE_ UNREGISTER_FAILED	Ошибка удаления регистрации "Jinn-Client"
0xC65007D4 В EventLog MessageId=2004	E_JINNSERVICE_START_ FAILED	Ошибка инициализации "Jinn-Client"
Создание объекта криптоконтекста		
0xC6500009	E_JS_CREATE_XML_ CRYPTOCONTEXT_LIGHT_ FAILED	Не удалось создать Light-реализацию криптографического контекста для работы с XML-документами
0xC650000A	E_JS_CREATE_XML_ CRYPTOCONTEXT_TE_ FAILED	Не удалось создать TE-реализацию криптографического контекста для работы с XML-документами

Код ошибки	Символьное представление в C++	Описание
0xC650000B	E_JS_CREATE_ BIBARYBASE64_ CRYPTOCONTEXT_LIGHT_ FAILED	Не удалось создать Light-реализацию криптографического контекста для работы с двоичными данными в формате Base64
0xC650000C	E_JS_CREATE_ BIBARYBASE64_ CRYPTOCONTEXT_TE_ FAILED	Не удалось создать TE-реализацию криптографического контекста для работы с двоичными данными в формате Base64
0xC650000D	E_JS_CREATE_BINARY_ CRYPTOCONTEXT_LIGHT_ FAILED	Не удалось создать Light-реализацию криптографического контекста для работы с двоичными данными
0xC650000E	E_JS_CREATE_BINARY_ CRYPTOCONTEXT_TE_ FAILED	Не удалось создать TE-реализацию криптографического контекста для работы с двоичными данными
Ошибки при обращении к методам криптоконтекстов		
0xC650001E	E_JS_COM_GET_ PROPERTY_ARGUMENT_ INVALIDE_POINTER	Для получения значения свойства криптографического контекста передан недействительный указатель
0xC6500021	E_JS_COM_METHOD_ ARGUMENT_INVALIDE_ POINTER	В метод криптографического контекста передан недействительный указатель
0xC6500022	E_JS_COM_METHOD_ ARGUMENT_BLANK_ STRING	Строка, переданная в метод криптографического контекста, не должна быть пустой
0xC6500023	E_JS_COM_TRANSFORM_ RESULT_IS_EMPTY	Результат XSLT-преобразования — пустая строка. Подпись невозможна
Ошибки методов подписания документа и проверки подписи		
0xC6500031	E_JS_TOKEN_OPEN_ FAILED	Не удалось открыть ключевой носитель
0xC6500032	E_JS_CRYPTOCONTAINER_ OPEN_FAILED	Не удалось открыть криптографический контейнер
0xC6500033	E_JS_CERTIFICATE_OPEN_ FAILED	Не удалось открыть сертификат
0xC6500034	E_JS_XSLT_FAILED	Ошибка XSLT-преобразования
0xC6500035	E_JS_SIGN_DOCUMENT_ FAILED	Ошибка формирования электронной подписи документа
0xC6500036	E_JS_CHECK_SIGNATURE_ FAILED	Ошибка проверки электронной подписи документа
0xC6500037	E_JS_REFUSED_TO_SIGN	Операция формирования электронной подписи документа прервана пользователем
0xC6500038	E_JS_DOCUMENT_ CONVERT_FAILED	Ошибка преобразования формата документа
0xC6500039	E_JS_REFUSED_TO_ CHECKSIGNATURE	Операция проверки электронной подписи документа прервана
0xC650003A	E_JS_ERROR_WRITE_ CONTAINER	Ошибка записи криптографического контейнера
0xC650003B	E_JS_ERROR_READ_ TOKEN	Ошибка чтения ключевого носителя
0xC650003C	E_JS_ERROR_WRITE_ TOKEN	Ошибка записи на ключевой носитель

Код ошибки	Символьное представление в C++	Описание
0xC650003D	E_JS_ERROR_INVALID_DATA	Неверные данные
0xC650003E	E_JS_ERROR_PARSE_CRYPTO_CONTAINER	Ошибка разбора криптографического контейнера
0xC650003F	E_JS_ERROR_PARSE_CERTIFICATE	Ошибка разбора сертификата
0xC6500040	E_JS_ERROR_UNKNOWN	Неопределенная ошибка
0xC6500041	E_JS_ERROR_NODE_NOT_FOUND	В XML-документе не найден элемент с указанным значением атрибута Id
0xC6500042	E_JS_ERROR_FEW_NODES_FOUND	В XML-документе найдено несколько элементов с указанным значением атрибута Id
0xC6500043	E_JS_ERROR_ATTRIBUTE_NOT_FOUND	В электронной подписи документа типа XADES в элементе Signature не найден атрибут Id
0xC6500044	E_JS_ERROR_OPERATION_ABORT	Операция отменена
0xC6500045	E_JS_ERROR_CERT_ANS_SECKEY_DISCREPANCY	Введен неверный пароль или сертификат не соответствует криптографическому контейнеру
0xC6500046	E_JS_ERROR_UPDATE_CRYPTO_CONTAINER	Ошибка обновления криптографического контейнера
0xC6500047	E_JINNSERVICE_INCORRECTPASSWD	Введен неверный пароль
0xC6500048	E_JS_DOCUMENT_READ_FAILED	Не удалось открыть документ
0xC6500049	E_JS_XSLT_READ_FAILED	Не удалось открыть XSLT-схему
0xC6500050	E_JS_STORE_TO_BUF_FAIL	Ошибка сохранения результата XSLT-преобразования в буфер
0xC6500053	E_CERTIFICATE_NOT_FOUND	Сертификат не найден
0xC6500054	E_CRYPTOCONTAINER_NOT_FOUND	Криптографический контейнер не найден
0xC6500055	E_GET_RNG_FAIL	Ошибка генерации случайного числа
0xC6500056	E_SIGNATURE_FAIL_PRNG_NOT_INITIALIZE	Ошибка создания подписи. Программный датчик случайных чисел не инициализирован
0xC6500057	E_SIGNATURE_FAIL	Ошибка создания подписи
0xC6500058	E_SIGNATURE_FAIL_ASN1_WRONG	Ошибка создания подписи. Входные данные имеют неправильную ASN.1-структуру
0xC6500059	E_JS_SIGN_ABORT	Операция формирования электронной подписи документа прервана
0xC6500060	E_JS_CHECKSIGNATURE_ABORT	Операция проверки электронной подписи документа прервана
0xC6500082	E_JS_UNKNOWN_KEY_LENGTH	Неизвестная длина открытого ключа
0xC6500100	E_JS_UNKNOWN_CRYPTOCONTAINER_TYPE	Выбран криптографический контейнер неизвестного типа
0xC6500101	E_JS_REGEX_PARSE_FAILED	Ошибка разбора регулярного выражения

Код ошибки	Символьное представление в C++	Описание
0xC6500102	E_JS_INVALID_TITLE_ FORMAT	Неверный формат элемента оглавления
0xC6500103	E_JS_TITLE_PARENTID_ ERROR	Найдены элементы оглавления с неправильной ссылкой в поле parentId
0xC6500104	E_JS_TITLE_ID_ERROR	Дублирование значения Id элемента оглавления
0xC6500105	E_JS_PASSWORD_ ATTEMPTS_FAILED	Превышено количество попыток ввода пароля
0xC6500106	E_JS_CERTTIFICATE_ BEGIN_VALIDITY_FAILED	Период действия сертификата еще не начался
0xC6500107	E_JS_CERTTIFICATE_ END_VALIDITY_FAILED	Период действия сертификата истек
0xC6500108	E_JS_CERTIFICATE_NOT_ DIGITAL_SIGNATURE	Сертификат не предназначен для формирования электронной подписи документа
0xC6500109	E_JS_CONTAINER_NOT_ FOUND	Неверно указан сертификат или пароль
0xC6500110	E_JS_PASSWORD_ CORRUPT	Ошибка получения сохраненного пароля. Пароль поврежден
0xC6500111	E_JS_DECODE_BASE_64	Ошибка декодирования значения в формате base64
0xC6500112	E_JS_XADES_CONSTANT_ NOT_FOUND	Константа Xades не найдена.%n Индекс константы
0xC6500113	E_JC_INVALIDE_CMS_ GOST_VERSION	Версия ГОСТ переданного сертификата не совпадает с версией ГОСТ CMS-контейнера, содержащего подпись
0xC6500114	E_JC_DECODE_ CERTIFICATE	Ошибка декодирования сертификата
0xC6500115	E_JC_EMPTY_CERTIFICATE	Сертификат пуст
0xC6500116	E_JS_CHECK_SIGNATURE_ WRONG	Подпись неверна
Ошибки модуля TECrypt		
0xC6600001	E_TE_HEADER_FILE_ASN_ ERR	Неверный формат заголовочного файла
0xC6600002	E_TE_KEY_FILE_ASN_ERR	Неверный формат ключевого файла
0xC6600003	E_TE_MASK_FILE_ASN_ ERR	Неверный формат файла масок
0xC6600004	E_TE_KEY_FILE_LEN_ERR	Неверная длина ключевого файла
0xC6600005	E_TE_MASK_FILE_LEN_ ERR	Неверная длина файла масок
0xC6600006	E_TE_MAC_MASK_FILE_ ERR	Ошибка целостности файла масок контейнера "КриптоПро"
0xC6600007	E_TE_CERTIFICATE_ERR	Ошибка сертификата X.509
0xC6600008	E_TE_SEC_KEY_ERR	Введен неверный пароль
0xC6600009	E_TE_PRNG_NOT_ INITIALIZE	Программный датчик случайных чисел не инициализирован
0xC6600030	E_TE_PKCS_15_FILE_ ASN_ERR	Контейнер не соответствует формату PKCS#15
0xC6600031	E_TE_PKCS_15_MAC_ERR	Ошибка целостности контейнера PKCS#15

Код ошибки	Символьное представление в C++	Описание
0xC6600032	E_TE_PKCS_15_PASS_ERR	Введен неверный пароль
0xC6600033	E_TE_PKCS_15_PRNG_ NOT_INITIALIZE	Программный датчик случайных чисел не инициализирован
Ошибки работы с libxml2		
0xC6500061	E_JS_CREATE_XMLDOC_ ROOTNODE_FAIL	Ошибка создания корневого узла XML-документа
0xC6500062	E_JS_CREATE_XMLDOC_ NAMESPACE_FAIL	Ошибка создания пространства имен в XML-документе
0xC6500063	E_JS_XMLDOC_SAVE_TO_ MEM_FAIL	Ошибка сохранения XML-документа в память
0xC6500064	E_JS_XMLDOC_ADD_ CHILD_NODE_FAIL	Ошибка сохранения XML-документа в память
0xC6500065	E_JS_XMLDOC_NOT_ INITIALIZE	XML-документ не инициализирован
0xC6500066	E_JS_READ_XMLDOC_ FROM_FILE_FAIL	Ошибка чтения XML-документа из файла
0xC6500067	E_JS_XMLDOC_ADD_ CHILD_FAIL	Ошибка добавления дочернего узла в XML-документ
0xC6500068	E_JS_CREATE_XMLDOC_ NODE_FAIL	Ошибка создания элемента XML-документа
0xC6500069	E_JS_CREATE_XMLDOC_ NO_NAMESPACE	Ошибка. Документ не содержит ожидаемого пространства имен
0xC6500070	E_JS_XMLDOC_ADD_ ATTRIBUTE_TO_NODE_ FAIL	Ошибка добавления атрибута к XML-элементу
0xC6500071	E_JS_XMLDOC_C14N_FAIL	Ошибка канонизации документа
0xC6500073	E_JS_XMLDOC_PARCE_ MEMORY_FAIL	Ошибка загрузки XML-документа из памяти
0xC6500074	E_JS_XMLDOC_ERROR_ WHILE_REGISTER_NS	Ошибка регистрации пространства имен для XPath-выражений
0xC6500075	E_JS_XMLDOC_INVALID_ NS_LIST_FORMAT	Переданный список пространств имен имеет неверный формат
0xC6500076	E_JS_XMLDOC_UNABLE_ REGISTER_NS	Невозможно зарегистрировать пространство имен
0xC6500077	E_JS_XMLDOC_FIND_ TARGET_NODE_FAIL	Ошибка поиска указанного элемента в XML-документе
0xC6500078	E_JS_XMLDOC_XPATH_ CONTEXT_CREATE_FAIL	Ошибка создания XPath-контекста
0xC6500079	E_JS_CHECK_SIGN_PASS_ EMPTY_DOC	Ошибка проверки электронной подписи документа.%n Передан пустой документ
0xC6500081	E_JS_XMLNODE_CREATE_ FAIL	Ошибка в библиотеке libxml2. Не удалось создать XML-элемент
0xC65000B1	E_JS_XMLNAME_WRONG_ OID	Имя пользовательского атрибута имеет неверный формат и не может быть использовано как имя XML-элемента
0xC65000B2	E_JS_WRONG_ ATTRIBUTE_STRING	Строка с пользовательскими атрибутами имеет неверный формат
Ошибки, связанные с доверенной средой		
0xC6500083	E_JS_TE_SIGN_FAIL	Ошибка формирования электронной подписи документа

Код ошибки	Символьное представление в C++	Описание
0xC6500084	E_JS_READ_CONFIG_ ERROR	Ошибка чтения файла настроек
0xC6500085	E_JS_HASH_SIGN_NOT_ SUPPORT_WHILE_ VISUALIZATION	Ошибка формирования электронной подписи документа.%n Формирование подписи для хэш-кода документа в режиме с обязательной визуализацией не поддерживается
0xC65000AF	E_JS_TE_INIT_PRNG_FAIL	Ошибка инициализации случайного числа в доверенной среде
Ошибки проверки атрибутов при формировании CMS-подписи		
0xC6500086	E_JS_ATTRIBUTE_ CONTENTTYPE_FOUND	В строке подписываемых атрибутов не должно быть атрибута contentType
0xC6500087	E_JS_ATTRIBUTE_ MESSAGEDIGEST_FOUND	В строке подписываемых атрибутов не должно быть атрибута MessageDigest
0xC6500088	E_JS_ATTRIBUTE_BOTH_ FOUND	В строке подписываемых атрибутов не должно быть атрибутов contentType и MessageDigest
0xC6500089	E_JS_CHECK_ATTRIBUTE_ FAIL	Функция проверки атрибутов вернула неизвестное значение
Ошибки проверки целостности		
0xC6500093	E_JS_CHECK_LAUNCH_ ICHECK_FAIL	Не удалось запустить утилиту проверки целостности
0xC6500094	E_JS_CHECK_INTEGRITY_ FAIL	Ошибка при проверке целостности файлов
0xC6500096	E_JS_TE_LOAD_WRONG_ SOURSE	"Jinn-Client" не может продолжить работу. В текущей конфигурации доверенная среда может быть загружена только с ПАК "Соболь"
0xC65000B0	E_JS_NO_SABLE	ПАК "Соболь" не установлен, приложение будет закрыто

Пример обработки исключений в JavaScript

```
//В JavaScript необходимо объявить "константы" с кодами ошибок
var E_JS_COM_METHOD_ARGUMENT_BLANK_STRING = 0xC6500022;
try {
   var JinnService = new ActiveX-Object("JinnServiceLibrary.JinnService");
   var BinaryBase64CryptoContext = Jinn- Service.CreateBinaryBase64
CryptoContext();
   BinaryBase64CryptoContext.SetProperty("", "DocTitle");
}
catch (ex) {
   switch( 0xFFFFFFFF + ex.number + 1 )
{
case E_JS_COM_METHOD_ARGUMENT_BLANK_STRING:
   //Код обработки ошибки
   break;
default:
   //При выполнении скрипта в IE
   alert(ex.name + ": " + ex.message);
   //При выполнении скрипта в Microsoft Windows Base Script Host
   WScript.Echo(ex.name + ": " + ex.message);
   break;
}
}
```