Телефон в Ижевске: +7 (3412) 72-11-00
Телефон в Москве: +7 (499) 277-1560
Сайт: www.directum.ru
E-mail: support@directum.ru

DIRECTUM. Описание интерфейсов модулей расширения

Назначение документа

Документ содержит описание свойств и методов для разработки модулей расширения: проверки орфографии, шифрования и подписания, интеграции с системами мгновенных сообщений, наблюдения за открытыми документами. Предназначен для разработчиков, модифицирующих систему под бизнес-процессы организации.

Содержание

Термины и сокращения

Контакт
Участник информационного взаимодействия в системе мгновенных сообщений.

Хранилище сертификатов
Специальные файлы или каталоги операционной системы, в которых хранятся цифровые сертификаты.

Сессионный (сеансовый) ключ
Ключ, используемый абонентами в рамках одного сеанса общения (или выполнения операции). Применяется для шифрования открытого текста, а затем сессионный ключ шифруется открытым ключом получателя.

Общие положения

Интерфейсы модулей расширения объявлены в библиотеке SBPluginInterfaceLibrary.tlb. Библиотека с реализацией модуля расширения экспортирует функцию **GetInterface**, которая возвращает ссылку на реализацию модуля расширения. Имя библиотеки модуля расширения должно соответствовать маске SP*.dll.

Примечание

Для модулей расширения, написанных на платформе .NET, вместо экспортируемой функции **GetInterface** в сборке должен быть объявлен публичный класс со статическим методом **GetInterface**.

Библиотека модуля расширения должна быть расположена в папке с установленной клиентской частью системы DIRECTUM:

\Plugins\<Тип модуля расширения>\<Имя модуля расширения>\

- <Тип модуля расширения> имеет одно из значений:
  - **Encryption** – для модулей расширения шифрования и подписания;
  - **SpellCheck** – для модулей расширения проверки орфографии;
  - **IM** – для модулей интеграции с системами мгновенных сообщений;
  - **DocumentObserver** – для модулей наблюдения за открытыми документами;
  - **FileStorage** – для модулей расширения файлового хранилища.
- <Имя модуля расширения> – любое имя, уникальное в рамках директории.

Синтаксис функции **GetInterface**:

```
function GetInterface: IPlugin; stdcall;
```

Всеми модулями расширения должен быть реализован интерфейс IPlugin. Тогда как IPluginSettings является вспомогательным интерфейсом для работы с настройками модуля расширения.

Для создания модуля расширения необходимо реализовать также специфические интерфейсы. Подробнее см. в описании каждого из них:

- проверка орфографии

- шифрование и подписание
- наблюдение за открытыми документами
- интеграция с системами мгновенных сообщений

IPlugin – интерфейс модуля расширения

Синтаксис:

```
IPlugin = interface(IDispatch)
```

Методы

Метод	Описание
GetDefaultSettings	Возвращает настройки модуля расширения по умолчанию
Initialize	Инициализирует модуль расширения настройками пользователя
CheckSettings	Проверяет корректность переданных настроек
CheckEnvironment	Проверяет возможность работы модуля расширения на конкретном рабочем месте

Свойства

Свойство	Описание
PlatformVersion	Версия платформы, для которой разрабатывается модуль расширения
Name	Имя модуля расширения
Title	Заголовок модуля расширения
Description	Описание модуля расширения

GetDefaultSettings – получить настройки модуля расширения по умолчанию

Синтаксис:

```
procedure GetDefaultSettings(const Settings: IPluginSettings);
```

Параметры:

- *Settings* – настройки модуля расширения.

Возвращаемое значение:

Настройки модуля расширения по умолчанию.

Описание:

Если пользовательские настройки не были заданы, то в *Settings* передаются настройки модуля по умолчанию. Подробнее о работе с настройками см. описание интерфейса IPluginSettings.

Initialize – инициализировать модуль расширения в соответствии с настройками

Синтаксис

```
procedure Initialize(const Settings: IPluginSettings);
```

Описание

Метод использует пользовательские настройки (если не заданы, то настройки по умолчанию) и инициализирует модуль расширения. Подробнее о работе с настройками см. описание интерфейса IPluginSettings.

CheckSettings – проверить корректность переданных настроек

Синтаксис:

```
procedure CheckSettings(const Settings: IPluginSettings);
```

Описание:

Метод вызывается при изменении пользовательских настроек. Если переданные настройки некорректны, то генерируется исключение. Подробнее о работе с настройками см. описание интерфейса IPluginSettings.

CheckEnvironment – проверить возможность работы модуля расширения на конкретном рабочем месте

Синтаксис:

```
function CheckEnvironment: WordBool;
```

Возвращаемое значение:

True, если модуль расширения может функционировать на конкретном рабочем месте, иначе **False**. При отсутствии возможности использования модуль расширения будет выгружен из оперативной памяти и не будет отображаться в окне параметров системы в списке модулей расширения.

Описание:

Метод вызывается для проверки возможности работы на текущем рабочем месте. Обязателен для выполнения перед началом работы на любом рабочем месте.

PlatformVersion – версия платформы, для которой разрабатывался модуль расширения

Синтаксис:

```
property PlatformVersion: WideString read Get_PlatformVersion;
```

Возвращаемое значение:

Версия платформы, для которой разрабатывался модуль расширения. Версия указывается с точностью до редакции релиза. Например, 7.12.0.

Описание:

Для версий, выпущенных раньше, модуль расширения загружен не будет. Для более поздних версий платформы модуль расширения будет загружен, если в платформу не были внесены критичные изменения, влияющие на функционирование модуля расширения.

Свойство используется только для чтения.

Name – имя модуля расширения

Синтаксис:

```
property Name: WideString read Get_Name;
```

Возвращаемое значение:

Уникальное имя модуля расширения. Например, GUID.

Описание:

Значение свойства используется только для уникальной идентификации модуля расширения в системе и доступно только для чтения.

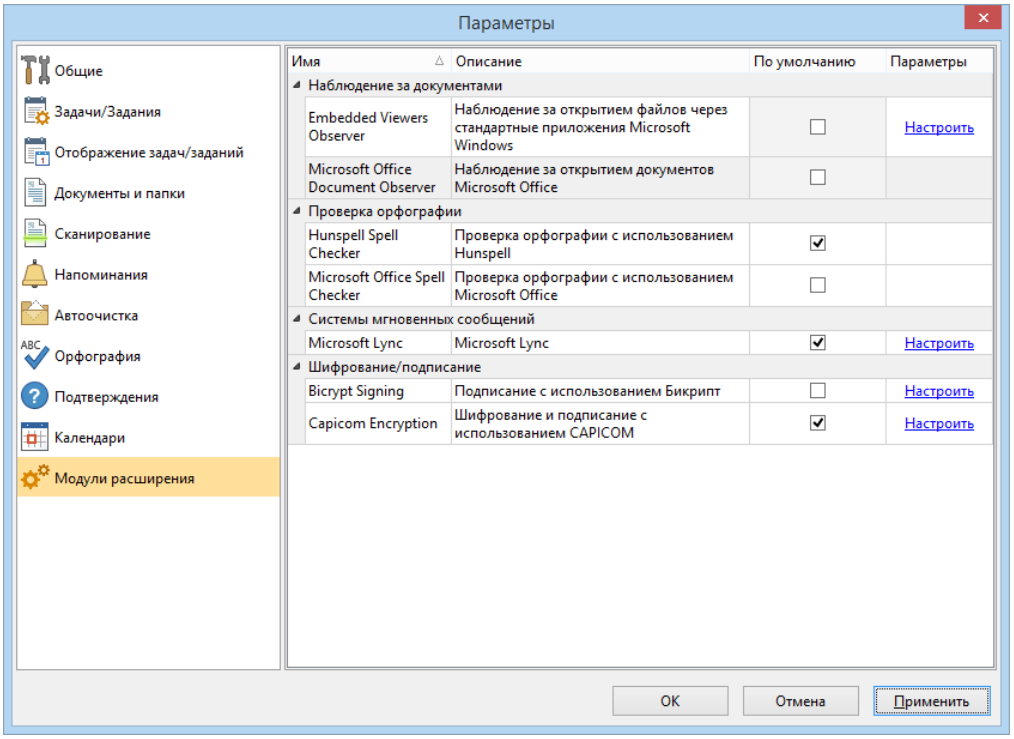
Title – заголовок модуля расширения

Синтаксис:

```
property Title: WideString read Get_Title;
```

Описание:

Заголовок модуля расширения отображается в списке модулей расширения в параметрах проводника системы. Свойство используется только для чтения.



Description – описание модуля расширения

Синтаксис:

```
property Description: WideString read Get_Description;
```

Описание:

Описание модуля расширения отображается в списке модулей расширения в параметрах проводника системы. Свойство используется только для чтения.

IPluginSettings – вспомогательный интерфейс для работы с настройками модулей расширения

Синтаксис:

```
IPluginSettings = interface(IDispatch)
```

Методы

Метод	Описание
AddSetting	Добавляет настройку в список настроек модуля расширения
ValueByName	Возвращает значение настройки по имени

Свойства

Свойство	Описание
Count	Количество настроек
Names	Имена настроек
ValueTypes	Типы данных настроек

AddSetting – добавить настройку в список настроек модулей расширения

Синтаксис:

```
procedure AddSetting(const Name: WideString; DefaultValue: OleVariant;
  ValueType: TSettingValueType; MainStorage: TSettingStorage);
```

Параметры:

- *Name* – имя настройки;
- *DefaultValue* – значение настройки по умолчанию;
- *ValueType* – тип данных настройки. Принимает одно из значений перечислимого типа **TSettingValueType**:
  - **svtBoolean** – логический тип;
  - **svtDouble** – числовой тип;
  - **svtString** – строковый тип;
  - **svtPassword** – строковый тип, при котором в форме настроек будет отображаться в виде символов «*».
- *MainStorage* – основное хранилище настроек. Принимает одно из значений перечислимого типа **TSettingStorage**:
  - **ssUserStorage** – для пользовательских настроек;
  - **ssServerStorage** – для серверных настроек.

Описание:

Метод добавляет настройку в список настроек. Используется в методе GetDefaultSettings интерфейса IPlugin.

ValueByName – получить значение настройки по имени

Синтаксис:

```
function ValueByName(const Name: WideString): OleVariant;
```

Параметры:

- *Name* – имя настройки.

Описание:

Метод возвращает текущее значение настройки по ее имени. Метод используется в методах Initialize и CheckSettings интерфейса IPlugin.

Count – количество настроек

Синтаксис:

```
property Count: Integer read Get_Count;
```

Описание:

Свойство возвращает количество настроек модуля расширения. Свойство используется только для чтения.

Names – имена настроек

Синтаксис:

```
property Names[Index: Integer]: WideString read Get_Names;
```

Описание:

Свойство возвращает имя настройки по ее индексу. Свойство используется только для чтения.

ValueTypes – типы данных настроек

Синтаксис:

```
property ValueTypes[Index: Integer]: TSettingValueType read Get_ValueTypes;
```

Описание:

Возвращает тип данных настройки по ее индексу. Принимает одно из значений перечислимого типа **TSettingValueType**:

- **svtBoolean** – логический тип;
- **svtDouble** – числовой тип;
- **svtString** – строковый тип;
- **svtPassword** – строковый тип, в форме настроек будет отображаться в виде символов «*».

Свойство используется только для чтения.

Модуль расширения проверки орфографии

Для создания модуля расширения для проверки орфографии необходимо реализовать интерфейсы ISpellCheckerPlugin и ISpellCheckerPlugin2. Вспомогательные интерфейсы ISpellingErrorList и ISuggestions реализовывать не нужно.

ISpellCheckerPlugin – интерфейс проверки орфографии

Синтаксис:

```
ISpellCheckerPlugin = interface(IPlugin)
```

Методы

Метод	Описание
CheckSpelling	Проверяет орфографию
GetSuggestions	Предлагает варианты замены слов, не найденных в стандартном словаре

CheckSpelling – проверить орфографию

Синтаксис:

```
procedure CheckSpelling(const Text: WideString; const ErrorList:
ISpellingErrorList);
```

Параметры:

- *Text* – исходный текст;
- *ErrorList* – список ошибочно написанных слов, найденных в тексте.

Описание:

Метод проверяет орфографию в тексте, переданном в параметре *Text*. Все найденные ошибки добавляет в список, переданный в параметре *ErrorList*. Подробнее о списке ошибок см. описание интерфейса ISpellingErrorList.

GetSuggestions – сформировать список вариантов замены слова

Синтаксис:

```
procedure GetSuggestions(const ErrorText: WideString; const Suggestions:
ISuggestions);
```

Параметры:

- *ErrorText* – слово, содержащее орфографическую ошибку;
- *Suggesions* – список вариантов замены слова.

Описание:

Метод предлагает варианты замены для ошибочно написанного слова, переданного в параметре *ErrorText*. Все варианты добавляются в список, переданный в параметре *Suggesions*. Подробнее о списке вариантов замены см. описание интерфейса ISuggestions.

ISpellCheckerPlugin2 – интерфейс проверки орфографии с использованием дополнительных словарей

Синтаксис:

```
ISpellCheckerPlugin2 = interface(ISpellCheckerPlugin)
```

Методы

Метод	Описание
AddSpelling	Добавляет слово в словарь проверки орфографии
DeleteSpelling	Удаляет слово из словаря проверки орфографии

Свойства

Свойство	Описание
CanAddSpelling	Признак возможности изменять состав слов в словаре

AddSpelling – добавить слово в словарь проверки орфографии

Синтаксис:

```
procedure AddSpelling(const Text: WideString);
```

Параметры:

- *Text* – новое слово.

Описание:

Метод добавляет слово, переданное в параметре *Text*, в словарь проверки орфографии. При последующей проверке орфографии добавленное слово не должно считаться ошибочным.

DeleteSpelling – удалить слово из словаря проверки орфографии

Синтаксис:

```
procedure DeleteSpelling(const Text: WideString);
```

Параметры:

- *Text* – слово, исключаемое из словаря.

Описание:

Метод удаляет слово, переданное в параметре *Text*, из словаря проверки орфографии. При последующей проверке орфографии удаленное слово будет считаться ошибочным.

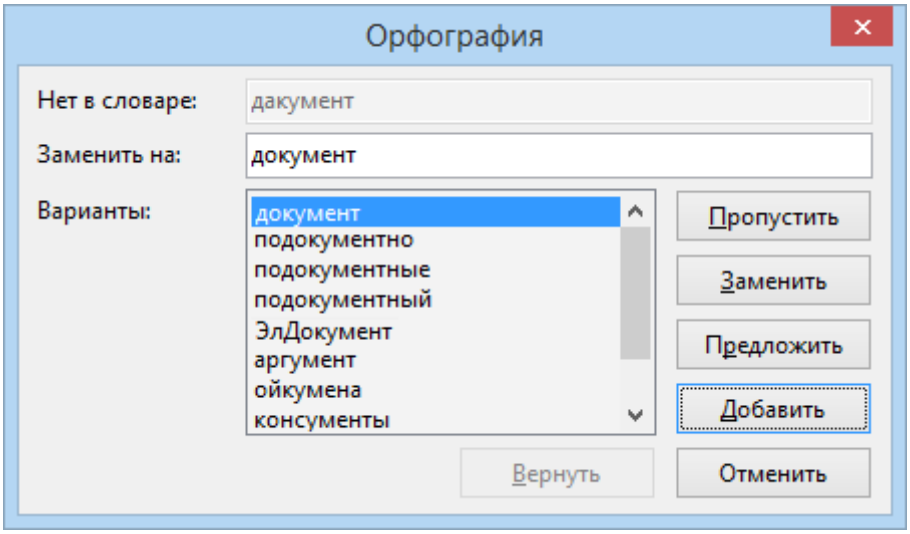
CanAddSpelling – признак возможности изменять состав слов в словаре

Синтаксис:

```
property CanAddSpelling: WordBool read Get_CanAddSpelling;
```

Возвращаемое значение:

True, если добавление и удаление слов словаря доступно, иначе **False**. Если состав слов словаря можно изменять, то в окне проверки орфографии будет доступна кнопка **Добавить**:



Описание:

Свойство указывает на наличие возможности добавлять и удалять слова из словаря проверки орфографии. Свойство используется только для чтения.

ISpellingErrorList – вспомогательный интерфейс для работы со списком ошибок проверки орфографии

Синтаксис:

```
ISpellingErrorList = interface(IDispatch)
```

Методы

Метод	Описание
Add	Добавляет слово в список ошибок при проверке орфографии

Add – добавить слово в список ошибок при проверке орфографии

Синтаксис:

```
procedure Add(const ErrorText: WideString; ErrorStartPos: Integer;
ErrorEndPos: Integer);
```

Параметры:

- *ErrorText* – слово, написанное с ошибкой;
- *ErrorStartPos*, *ErrorEndPos* – номера позиций начала и окончания ошибочного слова в проверяемом тексте.

Описание:

Метод добавляет слово, написанное неверно, в список ошибок при проверке орфографии.

ISuggestions – вспомогательный интерфейс для работы со списком вариантов замены ошибочного слова

Синтаксис:

```
ISuggestions = interface(IDispatch)
```

Методы

Метод	Описание
Add	Добавить слово в список вариантов замен ошибочного слова

Add – добавить альтернативный вариант написания слова

Синтаксис:

```
procedure Add(const Value: WideString);
```

Параметры:

- *Value* – альтернативный вариант написания слова.

Описание:

Метод добавляет вариант замены, переданный в параметре *Value*, в список вариантов замен ошибочно написанного слова.

Модуль расширения шифрования и подписания

Для создания модуля расширения шифрования и подписания необходимо реализовать интерфейсы IEncryptionPlugin, ICertificateStorage, ICertificate, а также ICertificate2, если требуется проверка действительности сертификата ЭП.

При этом сертификатом считается логический сертификат модуля расширения – класс, реализующий интерфейс ICertificate. Тогда как хранилище сертификатов понимается как логическое хранилище сертификатов модуля расширения – класс, реализующий интерфейс ICertificateStorage.

Если необходимо передавать в модуль расширения шифрования и подписания дополнительную информацию, например, подразделение пользователя, реализуйте интерфейс IEncryptionPlugin2.

Если необходимо отображать эту дополнительную информацию в окне «Подписи документа», реализуйте интерфейс IAdditionalInfoList.

IEncryptionPlugin – интерфейс шифрования и подписания

Синтаксис:

```
IEncryptionPlugin = interface(IPlugin)
```

Методы

Метод	Описание
CreateSignature	Создает ЭП
Decrypt	Расшифровывает текст, зашифрованный паролем
DecryptWithCertificate	Расшифровывает текст, зашифрованный открытым ключом сертификата
Encrypt	Зашифровывает текст паролем
EncryptWithCertificate	Зашифровывает текст открытым ключом сертификата
GetCertificate	Возвращает пустой сертификат
GetCertificateStorage	Возвращает хранилище сертификатов
GetSessionKey	Возвращает случайный сессионный ключ
Hash	Возвращает хэш переданного текста
VerifySignature	Проверяет достоверность ЭП

Свойства

Свойство	Описание
IsCertificateStorageSupported	Признак использования хранилища сертификатов
CanEncrypt	Признак возможности шифрования

CreateSignature – создать ЭП

Синтаксис:

```
function CreateSignature(const Content: WideString; const
CreateCertificate: ICertificate;
  out Signature: WideString; out CreateMsg: WideString;
  var CreateDateTime: TDateTime): WordBool;
```

Параметры:

- *Content* – исходный текст;
- *CreateCertificate* – сертификат ЭП;
- *Signature* – ЭП;
- *CreateMsg* – текст ошибки, возникшей при получении подписи;
- *CreateDataTime* – время создания подписи. Данный параметр может быть изменен, например при использовании службы штампов времени.

Возвращаемое значение:

- **True**, если подпись была получена успешно. При этом в параметр *Signature* записывается полученная подпись;
- **False**, если при получении подписи возникли ошибки. При этом в параметр *CreateMsg* записывается текст возникшей ошибки, который будет отображен пользователю.

Описание:

Метод получает ЭП для текста, переданного в параметре *Content*, с помощью сертификата, переданного в параметре *CreateCertificate*. Подробнее о работе с сертификатами ЭП см. описание интерфейса ICertificate.

Decrypt – расшифровать текст, зашифрованный паролем

Синтаксис:

```
function Decrypt(const Source: WideString; const Password: WideString):
WideString;
```

Параметры:

- *Source* – зашифрованный паролем текст;
- *Password* – пароль.

Возвращаемое значение:

Расшифрованный текст.

Описание:

Метод расшифровывает текст, переданный в параметре *Source*, с помощью пароля, переданного в параметре *Password*.

DecryptWithCertificate – расшифровать текст, зашифрованный открытым ключом сертификата

Синтаксис:

```
function DecryptWithCertificate(const Source: WideString): WideString;
```

Параметры:

- *Source* – зашифрованный текст.

Описание:

Метод расшифровывает текст, переданный в параметре *Source*, с помощью закрытого ключа сертификата. Сертификат, с помощью которого необходимо расшифровать текст, определяется модулем самостоятельно.

Encrypt – зашифровать текст паролем

Синтаксис:

```
function Encrypt(const Source: WideString; const Password: WideString):
WideString;
```

Параметры:

- *Source* – текст;
- *Password* – пароль.

Возвращаемое значение:

Зашифрованный текст.

Описание:

Метод зашифровывает текст, переданный в параметре *Source*, с помощью пароля, переданного в параметре *Password*.

EncryptWithCertificate – зашифровать текст с помощью открытого ключа сертификата

Синтаксис:

```
function EncryptWithCertificate(const Source: WideString;
  const Certificate: ICertificate): WideString;
```

Параметры:

- *Source* – текст;
- *Certificate* – открытый ключ сертификата.

Возвращаемое значение:

Зашифрованный текст.

Описание:

Метод зашифровывает текст, переданный в параметре *Source*, с помощью открытого ключа сертификата, переданного в параметре *Certificate*. Подробнее о работе с сертификатом см. описание интерфейса ICertificate.

GetCertificate – получить пустой сертификат

Синтаксис:

```
function GetCertificate: ICertificate;
```

Возвращаемое значение:

Новый экземпляр класса, реализующего интерфейс ICertificate.

Описание:

Метод создает пустой сертификат – класс, реализующий интерфейс ICertificate, и возвращает его. Подробнее о работе с сертификатом см. описание интерфейса ICertificate.

GetCertificateStorage – получить хранилище сертификатов

Синтаксис:

```
function GetCertificateStorage: ICertificateStorage;
```

Описание:

Метод создает хранилище сертификатов, откуда будет производиться выбор и поиск сертификатов, и возвращает его. Подробнее о работе с хранилищем сертификатов см. описание интерфейса ICertificateStorage.

GetSessionKey – получить сессионный ключ

Синтаксис:

```
function GetSessionKey: WideString;
```

Описание:

Метод генерирует и возвращает случайный сессионный ключ.

Hash – вычислить хэш-код переданного текста

Синтаксис:

```
function Hash(const Source: WideString): WideString;
```

Параметры:

- *Source* – текст.

Возвращаемое значение:

Хэш-код строки, переданной в параметре *Source*.

VerifySignature – проверить достоверность ЭП

Синтаксис:

```
function VerifySignature(const Content: WideString; const Signature:
WideString;
  out VerifyCertificate: ICertificate; out VerifyMsg: WideString;
  out SignDate: TDateTime): WordBool;
```

Параметры:

- *Content* – текст;
- *Signature* – электронная подпись;
- *VerifyCertificate* – сертификат ЭП;
- *SignDate* – дата подписания текста;
- *VerifyMsg* – причина, по которой достоверность подписи определить не удалось.

Возвращаемое значение:

- **True**, если подпись достоверна. При этом в параметр *VerifyCertificate* записывается сертификат ЭП;
- **False**, если подпись не достоверна или достоверность подписи определить не удалось. При этом в параметр *VerifyMsg* записывается текст возникшей ошибки, который будет отображен пользователю.

Описание:

Метод проверяет достоверность электронной подписи, переданной в параметре *Signature*, для текста, переданного в параметре *Content*. Подробнее о работе с сертификатом см. описание интерфейса ICertificate.

IsCertificateStorageSupported – признак использования хранилища сертификатов

Синтаксис:

```
property IsCertificateStorageSupported: WordBool read
Get_IsCertificateStorageSupported;
```

Возвращаемое значение:

True, если обеспечена работа модуля расширения с использованием хранилища сертификатов, иначе **False**. В этом случае реализация интерфейса ICertificateStorage и метода GetCertificateStorage необязательна.

Описание:

Свойство определяет поддержку работы с хранилищами сертификатов. Свойство используется только для чтения.

CanEncrypt – признак применения шифрования

Синтаксис:

```
property CanEncrypt: WordBool read Get_CanEncrypt;
```

Возвращаемое значение:

True, если обеспечена возможность шифрования, иначе **False**. В этом случае реализация методов Decrypt, DecryptWithCertificate, Encrypt, EncryptWithCertificate, GetSessionKey необязательна.

Описание:
Свойство определяет поддержку шифрования. Свойство используется только для чтения.

ICertificateStorage – интерфейс хранилища сертификатов

Интерфейс должен реализовывать один из внутренних классов в модуле расширения, но не сам класс, реализующий интерфейс IPlugin и IEncryptionPlugin.

Синтаксис:

```
ICertificateStorage = interface(IDispatch)
```

Методы

Метод	Описание
Find	Определяет наличие сертификата в хранилище

Select	Показывает окно выбора сертификата в хранилище
Export	Экспортирует сертификат из хранилища

Свойства:

Свойство	Описание
Count	Количество сертификатов в хранилище

Find – определить наличие сертификата в хранилище

Синтаксис:

```
function Find(const CertificateID: WideString): WordBool;
```

Параметры:

- *CertificateID* – ИД сертификата.

Возвращаемое значение:

True, если сертификат есть в хранилище, иначе **False**.

Описание:

Метод определяет наличие сертификата в хранилище по ИД.

Select – выбрать сертификат в хранилище

Синтаксис:

```
procedure Select(const Title: WideString; const DisplayString: WideString;
  out CertificateID: WideString);
```

Параметры:

- *Title* – заголовок окна выбора сертификата;
- *DisplayString* – текст подсказки для пользователя о целях выбора сертификата;
- *CertificateID* – ИД выбранного сертификата.

Возвращаемое значение:

True, если сертификат был выбран, иначе **False**.

Описание:

Метод отображает окно выбора сертификата из хранилища.

Export – экспортировать сертификат из хранилища

Синтаксис:

```
procedure Export(const CertificateID: WideString; const FileName:
WideString;
  const Password: WideString; SavePrivateKey: WordBool);
```

Параметры:

- *CertificateID* – ИД сертификата;
- *FileName* – файл, в который будет экспортирован сертификат;
- *Password* – пароль. Используется, если для экспорта сертификата необходимо ввести пароль;
- *SavePrivateKey* – признак экспорта закрытого ключа сертификата.

Count – количество сертификатов в хранилище

Синтаксис:

```
property Count: Integer read Get_Count;
```

ICertificate – интерфейс сертификата ЭП

Интерфейс должен реализовывать один из внутренних классов в модуле расширения, но не сам класс, реализующий интерфейс IPlugin и IEncryptionPlugin.

Синтаксис:

```
ICertificate = interface(IDispatch)
```

Методы

Метод	Описание
IsValid	Определяет, действителен ли сертификат
LoadPublicKeyFromFile	Загружает в сертификат открытый ключ из файла
LoadPublicKeyFromStorage	Загружает в сертификат открытый ключ из хранилища
Display	Отображает информацию о сертификате
Save	Экспортирует сертификат в файл
LoadPrivateKeyFromFile	Загружает в сертификат закрытый ключ из файла
LoadPrivateKeyFromStorage	Загружает в сертификат закрытый ключ из хранилища

Свойства

Свойство	Описание
CanLoadPublicKeyFromFile	Признак возможности загрузки открытого ключа из файла
CanLoadPublicKeyFromStorage	Признак возможности загрузки открытого ключа из хранилища
CanLoadPrivateKeyFromFile	Признак возможности загрузки закрытого ключа из файла
CanLoadPrivateKeyFromStorage	Признак возможности загрузки закрытого ключа из хранилища
HasPrivateKey	Признак наличия закрытого ключа у сертификата
EncryptionAllowed	Признак использования сертификата для шифрования
SigningAllowed	Признак использования сертификата для подписания
CertificateID	Идентификатор сертификата
IssuerName	Издатель сертификата
PluginName	Имя модуля расширения
SerialNumber	Серийный номер сертификата

SubjectName	Владелец сертификата
ValidFromDate	Дата начала действия сертификата
ValidToDate	Дата окончания действия сертификата
Version	Версия сертификата
KeyFileExtensionFilter	Фильтр по расширению файлов для файлов с ключами сертификата

IsValid – определить, действителен ли сертификат

Синтаксис:

```
function IsValid(VerificationDate: TDateTime; NeedCheckDateValidity:
WordBool): WordBool;
```

Параметры:

- *VerificationDate* – дата подписи с использованием данного сертификата;
- *NeedCheckDateValidity* – признак необходимости проверки действительности сертификата.

Описание:

Метод определяет, действителен ли сертификат. Если параметр *NeedCheckDateValidity* равен **True**, то действительность сертификата проверяется на дату, переданную в параметре *VerificationDate*.

LoadPublicKeyFromFile – загрузить в сертификат открытый ключ из файла

Синтаксис:

```
function LoadPublicKeyFromFile(const FileName: WideString; const Password:
WideString;
  out LoadErrorMessage: WideString): TCertificateLoadPublicKeyResult;
```

Параметры:

- *FileName* – имя файла;
- *Password* – пароль, используемый при загрузке ключа;
- *LoadErrorMessage* – текст сообщения о результатах загрузки ключа.

Возвращаемое значение:

Одно из значений, определенных в перечислении **TCertificateLoadPublicKeyResult**:

- **lrSuccess** – загрузка прошла успешно;
- **lrInvalidPassword** – пароль не передан, но необходим, или передан неверный пароль;
- **lrNotFound** – файл не найден;
- **lrInvalidFile** – передан файл неверного формата;
- **lrUnknownError** – для всех остальных ошибок.

LoadPublicKeyFromStorage – загрузить в сертификат открытый ключ из хранилища

Синтаксис:

```
function LoadPublicKeyFromStorage(const CertificateID: WideString;
  out LoadErrorMessage: WideString): TCertificateLoadPublicKeyResult;
```

Параметры:

- *CertificateID* – ИД загружаемого сертификата;
- *LoadErrorMessage* – текст сообщения о результатах загрузки ключа.

Возвращаемое значение:

Одно из значений, определенных в перечислении **TCertificateLoadPublicKeyResult**:

- **lrSuccess** – загрузка прошла успешно;
- **lrUnknownError** – загрузка прошла с ошибками.

Display – отобразить информацию о сертификате

Синтаксис:

```
procedure Display;
```

Описание:

Метод отображает информацию о сертификате в отдельном окне.

Save – экспортировать сертификат в файл

Синтаксис:

```
procedure Save(const FileName: WideString; const Password: WideString;
  SavePrivateKey: WordBool);
```

Параметры:

- *FileName* – файл, в который будет экспортирован сертификат;
- *Password* – пароль;
- *SavePrivateKey* – признак необходимости экспорта закрытого ключа.

LoadPrivateKeyFromFile – загрузить закрытый ключ из файла в сертификат

Синтаксис:

```
function LoadPrivateKeyFromFile(const FileName: WideString; const Password:
WideString;
  out LoadErrorMessage: WideString): TCertificateLoadPrivateKeyResult;
```

Параметры:

- *FileName* – имя файла;

- *Password* – пароль, используемый при загрузке ключа;
- *LoadErrorMessage* – текст сообщения о результатах загрузки ключа.

Возвращаемое значение:

Одно из значений, определенных в перечислении **TCertificateLoadPrivateKeyResult**:

- **lpkrSuccess** – загрузка прошла успешно;
- **lpkrNeedFileName** – не передано имя файла;
- **lpkrInvalidPassword** – пароль не передан, но необходим, или передан неверный пароль;
- **lpkrIllegalCertificateID** – в файле указан неверный *CertificateID*;
- **lpkrNotFound** – если файл не найден;
- **lpkrInvalidFile** – если передан файл неверного формата;
- **lpkrUnknownError** – для всех остальных ошибок.

LoadPrivateKeyFromStorage – загрузить закрытый ключ в сертификат из хранилища

Синтаксис:

```
function LoadPrivateKeyFromStorage(const CertificateID: WideString;
  out LoadErrorMessage: WideString): TCertificateLoadPrivateKeyResult;
```

Параметры:

- *CertificateID* – ИД сертификата;
- *LoadErrorMessage* – текст сообщения о результатах загрузки ключа.

Возвращаемое значение:

Одно из значений, определенных в перечислении **TCertificateLoadPrivateKeyResult**:

- **lpkrSuccess** – загрузка прошла успешно;
- **lpkrIllegalCertificateID** – передан неверный *CertificateID*;
- **lpkrUnknownError** – для всех остальных ошибок.

CanLoadPublicKeyFromFile – признак возможности загрузки открытого ключа из файла

Синтаксис:

```
property CanLoadPublicKeyFromFile: WordBool read
Get_CanLoadPublicKeyFromFile;
```

Описание:

Свойство определяет, поддерживает ли сертификат возможность загрузки открытого ключа из файла.

Если значение свойства равно **False**, то реализация метода LoadPublicKeyFromFile не обязательна. Свойство используется только для чтения.

CanLoadPublicKeyFromStorage – признак возможности загрузки открытого ключа из хранилища

Синтаксис:

```
property CanLoadPublicKeyFromStorage: WordBool read
Get_CanLoadPublicKeyFromStorage;
```

Описание:

Свойство определяет, поддерживает ли сертификат возможность загрузки открытого ключа из хранилища.

Если значение свойства равно **False**, то реализация метода LoadPublicKeyFromStorage не обязательна. Свойство используется только для чтения.

CanLoadPrivateKeyFromFile – признак возможности загрузки закрытого ключа из файла

Синтаксис:

```
property CanLoadPrivateKeyFromFile: WordBool read
Get_CanLoadPrivateKeyFromFile;
```

Описание:

Свойство определяет, поддерживает ли сертификат возможность загрузки закрытого ключа из файла. Если значение свойства равно **False**, то реализация метода LoadPrivateKeyFromFile не обязательна. Свойство используется только для чтения.

CanLoadPrivateKeyFromStorage – признак возможности загрузки закрытого ключа из хранилища

Синтаксис:

```
property CanLoadPrivateKeyFromStorage: WordBool read
Get_CanLoadPrivateKeyFromStorage;
```

Описание:

Свойство определяет, поддерживает ли сертификат возможность загрузки закрытого ключа из хранилища. Если значение свойства равно **False**, то реализация метода LoadPrivateKeyFromStorage не обязательна. Свойство используется только для чтения.

HasPrivateKey – признак наличия закрытого ключа у сертификата

Синтаксис:

```
property HasPrivateKey: WordBool read Get_HasPrivateKey;
```

Описание:

Значение свойства равно **True** при наличии закрытого ключа сертификата, иначе **False**.

Описание:

Свойство определяет, загружен ли закрытый ключ в сертификат. Свойство используется только для чтения.

EncryptionAllowed – признак использования сертификата для шифрования

Синтаксис:

```
property EncryptionAllowed: WordBool read Get_EncryptionAllowed;
```

Описание:

Значение свойства равно **True** при использовании сертификатов для шифрования, иначе **False**.

SigningAllowed – признак использования сертификата для подписания

Синтаксис:

```
property SigningAllowed: WordBool read Get_SigningAllowed;
```

Описание:

Значение свойства равно **True** при использовании сертификата ЭП, иначе **False**.

CertificateID – ИД сертификата

Синтаксис:

```
property CertificateID: WideString read Get_CertificateID;
```

Описание:

Свойство возвращает уникальный идентификатор сертификата. Например, для сертификатов X.509 это может быть отпечаток.

IssuerName – издатель сертификата

Синтаксис:

```
property IssuerName: WideString read Get_IssuerName;
```

PluginName – имя модуля расширения

Синтаксис:

```
property PluginName: WideString read Get_PluginName;
```

Описание:

Свойство возвращает имя модуля расширения, которому принадлежит сертификат. Должен совпадать со значением свойства *Name* модуля расширения.

SerialNumber – серийный номер сертификата

Синтаксис:

```
property SerialNumber: WideString read Get_SerialNumber;
```

Описание:

Свойство возвращает серийный номер сертификата. Свойство используется только для чтения.

SubjectName – владелец сертификата

Синтаксис:

```
property SubjectName: WideString read Get_SubjectName;
```

Описание:

Свойство возвращает имя владельца сертификата. Свойство используется только для чтения.

ValidFromDate – дата начала действия сертификата

Синтаксис:

```
property ValidFromDate: TDateTime read Get_ValidFromDate;
```

ValidToDate – дата окончания действия сертификата

Синтаксис:

```
property ValidToDate: TDateTime read Get_ValidToDate;
```

Version – версия сертификата

Синтаксис:

```
property Version: Integer read Get_Version;
```

KeyFileExtensionFilter – фильтр по расширению файлов для файлов с ключами сертификата

Синтаксис:

```
property KeyFileExtensionFilter: WideString read
Get_KeyFileExtensionFilter;
```

Описание:

Свойство возвращает фильтр по расширениям файлов для ключей сертификата. Свойство используется для диалога открытия файла при выборе файла для загрузки открытого ключа из файла.

ICertificate2 – вспомогательный интерфейс шифрования и подписания

Интерфейс должен реализовывать тот же класс, что и ICertificate, но могут дополнительно использоваться и другие классы, не реализующие интерфейс **ICertificate**.

Интерфейс является необязательным к реализации. При его отсутствии работа производится с интерфейсом ICertificate.

Синтаксис:

```
ICertificate2 = interface(ICertificate)
```

Методы

Метод	Описание
IsValidEx	Определяет, действителен ли сертификат

IsValidEx – определить, действителен ли сертификат

Синтаксис:

```
function IsValidEx(VerificationDate: TDateTime; NeedCheckDateValidity:
WordBool;out InvalidationReason: TCertificateInvalidationReason): WordBool;
```

Параметры:

- *VerificationDate* – дата подписи с использованием данного сертификата;
- *NeedCheckDateValidity* – признак необходимости проверки действительности сертификата.

Описание:

Метод определяет, действителен ли сертификат. Если параметр *NeedCheckDateValidity* равен **True**, то действительность сертификата проверяется на дату, переданную в параметре *VerificationDate*.

Возвращаемое значение:

Одно из значений, определенных в перечислении **TCertificateInvalidationReason**:

- **cirCommon** – общая причина недействительности (подробности недоступны);
- **cirRevoked** – истек срок действия сертификата или сертификат отозван.

IAdditionalInfoList – интерфейс списка дополнительной информации о подписи

Интерфейс должен реализовывать один из внутренних классов в модуле расширения, но не сам класс, реализующий интерфейс IPlugin и IEncryptionPlugin.

Синтаксис:

```
IAdditionalInfoList = interface(IDispatch)
```

Свойства:

Свойство	Описание
Count	Количество строк дополнительной информации
Items	Строка дополнительной информации

Count – количество строк дополнительной информации

Синтаксис:

```
property Count: Integer read Get_Count;
```

Описание:
Свойство возвращает количество строк дополнительной информации.

Items – строка дополнительной информации

Синтаксис:

```
property Items[Index: Integer]: WideString read Get_Items;
```

Описание:
Свойство возвращает строку дополнительной информации по ее индексу.

IEncryptionPlugin2 – интерфейс шифрования и подписания

Синтаксис:

```
IEncryptionPlugin2 = interface(IEncryptionPlugin)
```

Методы

Метод	Описание
CreateSignature2	Создает ЭП
VerifySignature2	Проверяет достоверность ЭП

CreateSignature2 – создать ЭП

Синтаксис:

```
function CreateSignature2(
  const Content: WideString;
  const CreateCertificate: ICertificate;
  const AdditionalInfo: IAdditionalInfoList;
  out Signature: WideString;
  out CreateMsg: WideString;
  var CreateDateTime: TDateTime): WordBool;
```

Параметры:

- *Content* – исходный текст;

- *CreateCertificate* – сертификат ЭП;
- *AdditionalInfo* – дополнительная информация;
- *Signature* – ЭП;
- *CreateMsg* – текст ошибки, возникшей при получении подписи;
- *CreateDataTime* – время создания подписи. Данный параметр может быть изменен, например при использовании службы штампов времени.

Возвращаемое значение:

- **True**, если подпись была получена успешно. При этом в параметр *Signature* записывается полученная подпись;
- **False**, если при получении подписи возникли ошибки. При этом в параметр *CreateMsg* записывается текст возникшей ошибки, который будет отображен пользователю.

Описание:

Метод получает ЭП для текста *Content* с помощью сертификата *CreateCertificate*, и дополнительной информации *AdditionalInfo*. Подробнее о работе с сертификатами ЭП см. описание интерфейса ICertificate.

VerifySignature2 – проверить достоверность ЭП

Синтаксис:

```
function VerifySignature2(
  const Content: WideString;
  const Signature: WideString;
  out VerifyCertificate: ICertificate;
  out VerifyMsg: WideString;
  out SignDate: TDateTime;
  out AdditionalInfo: IAdditionalInfoList): WordBool;
```

Параметры:

- *Content* – текст;
- *Signature* – электронная подпись;
- *VerifyCertificate* – сертификат ЭП;
- *SignDate* – дата подписания текста;
- *VerifyMsg* – причина, по которой достоверность подписи определить не удалось.
- *AdditionalInfo* – список с дополнительной информацией.

Возвращаемое значение:

- **True**, если подпись достоверна. При этом в параметр *VerifyCertificate* записывается сертификат ЭП;
- **False**, если подпись не достоверна или достоверность подписи определить не удалось. При этом в параметр *VerifyMsg* записывается текст возникшей ошибки, который будет отображен пользователю.

Описание:

Метод проверяет достоверность электронной подписи *Signature* для текста *Content*. Подробнее о работе с сертификатом см. описание интерфейса ICertificate.

Модуль расширения интеграции с системами мгновенных сообщений

Для создания модуля расширения необходимо реализовать интерфейсы IMessagingPlugin, IContact, IStatus, IConversation.

Интерфейс **IGroup**, группа контактов системы мгновенных сообщений, устарел и используется только для обеспечения совместимости. Реализовывать данный интерфейс не требуется.

IMessagingPlugin – интерфейс интеграции с системами мгновенных сообщений

Синтаксис:

```
IMessagingPlugin = interface(IPlugin)
```

Методы

Метод	Описание
StartConversaton	Инициирует беседу (конференцию) со списком контактов
GetContact	Возвращает данные о контакте

Свойства

Свойство	Описание
Code	Код (типа) системы мгновенных сообщений
GroupCount	Количество групп контактов (устар.)
Group	Группа контактов (устар.)
Me	Контакт текущего пользователя
StatusCount	Количество статусов в системе мгновенных сообщений
Status	Статус в системе мгновенных сообщений
UnknownStatus	Статус «Состояние контакта неизвестно»

StartConversation – начать беседу (конференцию) со списком контактов

Синтаксис:

```
function StartConversation(ContactList: OleVariant): IConversation;
```

Параметры:

- *ContactList* – одно имя контакта (в виде строки) или массив строк для нескольких имен.

Описание:

Метод инициирует беседу с одним или несколькими контактами.

GetContact – получить данные о контакте

Синтаксис:

```
function GetContact(const UserName: WideString; IsOsAuth: WordBool):
IContact;
```

Параметры:

- *UserName* – имя пользователя Windows или логин SQL-сервера;
- *IsOsAuth* – признак аутентификации (Windows или SQL).

Описание:

Метод получает контакт по имени пользователя Windows при Windows-аутентификации или логин SQL-сервера при SQL-аутентификации.

Code – код типа системы мгновенных сообщений

Синтаксис:

```
property Code: WideString read Get_Code;
```

Описание:

Значение кода системы мгновенных сообщений, например «OCS», «SIP», «Jabber», «ICQ».

GroupCount – количество групп контактов

Синтаксис:

```
property GroupCount: Integer read Get_GroupCount;
```

Описание:

Свойство устарело и используется только для сохранения совместимости. Реализация данного свойства может возвращать **0**, либо безусловно генерировать исключение.

Group – группа контактов

Синтаксис:

```
property Group[Index: Integer]: IGroup read Get_Group;
```

Описание:

Свойство устарело и используется только для сохранения совместимости. Реализация данного свойства может возвращать **nil**, либо безусловно генерировать исключение.

Me – контакт текущего пользователя

Синтаксис:

```
property Me: IContact read Get_Me;
```

Описание:

Текущий пользователь определяется системой мгновенных сообщений самостоятельно и может не совпадать с текущим пользователем системы DIRECTUM, например в случае SQL-аутентификации.

StatusCount – количество статусов в системе мгновенных сообщений

Синтаксис:

```
property StatusCount: Integer read Get_StatusCount;
```

Status – статус в системе мгновенных сообщений

Синтаксис:

```
property Status[Index: Integer]: IStatus read Get_Status;
```

Описание:

Статусы имеют индексы от **0** до **StatusCount-1**. Данное свойство предназначено для получения полного перечня возможных статусов системы, а не для получения текущего статуса конкретного пользователя.

UnknownStatus – статус «Состояние контакта неизвестно»

Синтаксис:

```
property UnknownStatus: IStatus read Get_UnknownStatus;
```

Описание:

Данный статус будет возвращен для пользователей, информация о которых не может быть получена из системы мгновенных сообщений.

IContact – интерфейс контакта системы мгновенных сообщений

Методы

Метод	Описание
StartConversation	Инициировать беседу с контактом

Свойства

Свойство	Описание
Login	Логин контакта в системе мгновенных сообщений
DisplayName	Отображаемое имя контакта
Status	Статус контакта
Tagged	Признак «Оповещать об изменении состояния контакта»
Valid	Признак корректности контакта

StartConversation – инициирует беседу с контактом

Синтаксис:

```
function StartConversation: IConversation;
```

Login – логин контакта в системе мгновенных сообщений

Синтаксис:

```
property Login: WideString read Get_Login;
```

DisplayName – отображаемое имя контакта

Синтаксис:

```
property DisplayName: WideString read Get_DisplayName;
```

Status – статус контакта

Синтаксис:

```
property Status: IStatus read Get_Status;
```

Tagged – признак «Оповещать об изменении состояния контакта»

Синтаксис:

```
property Tagged: WordBool read Get_Tagged write Set_Tagged;
```

Описание:

Оповещение может выполняться как системой мгновенных сообщений, так и модулем расширения.

Valid – признак корректности контакта

Синтаксис:

```
property Valid: WordBool read Get_Valid;
```

Описание:

Если равен **False**, то значения остальных свойств не имеют смысла. Например, если запросить несуществующий контакт, то плагин возвращает контакт со значением свойства **Valid** равным **False**.

IStatus – интерфейс статуса контакта

Свойства

Свойство	Описание
ID	ИД статуса
DisplayName	Отображаемое имя статуса
Icon	Значок статуса

ID – ИД статуса

Синтаксис:

```
property ID: Integer read Get_ID;
```

Описание:

Платформой системы проверяется два статуса на равенство по ИД. Используется при обновлении статуса контакта в реальном времени.

DisplayName – отображаемое имя статуса

Синтаксис:

```
property DisplayName: WideString read Get_DisplayName;
```

Описание:

Значение может являться обычной строкой или строкой локализации в формате «Группа.КодСтроки».

Icon – значок статуса

Синтаксис:

```
property Icon: OleVariant read Get_Icon;
```

Описание:

Значение в виде массива байт.

IConversation – интерфейс беседы

Методы

Метод	Описание
SendMessage	Отправить сообщение

Свойства

Свойство	Описание
History	История сообщений в окне беседы (устар.)
Initialized	Признак того, что беседа инициализирована

SendMessage – отправляет сообщения

Синтаксис:

```
procedure SendMessage(const Message: WideString);
```

History – история сообщений в окне беседы

Синтаксис:

```
property History: WideString read Get_History;
```

Описание:

Свойство устарело и используется только для обеспечения совместимости. Реализация данного свойства может возвращать пустую строку, либо безусловно генерировать исключение.

Initialized – признак того, что беседа инициализирована

Синтаксис:

```
property Initialized: WordBool read Get_Initialized;
```

Описание:

Сообщение можно будет отправить только после установки свойства в значение **True**.

Модуль наблюдения за открытыми документами

Для создания модуля наблюдения за открытыми документами необходимо реализовать интерфейсы IDocumentObserverPlugin и IDocumentObserver.

IDocumentObserverPlugin – интерфейс модуля наблюдения за открытыми документами

Методы

Метод	Описание
SupportsExtension	Определяет, сможет ли плагин отследить закрытие файла
GetObserver	Получает наблюдателя, контролирующего состояние файла

SupportsExtension – определить возможность отследить закрытие файла

Синтаксис:

```
function SupportsExtension(const FileName: WideString): WordBool;
```

Параметры:

- *FileName* – наименование открытого файла.

Возвращаемое значение:

True, если отследить закрытие файла можно, иначе **False**.

GetObserver – создать наблюдателя, контролирующего состояние файла

Синтаксис:

```
function GetObserver(const FileName: WideString): IDocumentObserver;
```

Параметры:

- *FileName* – наименование открытого файла.

IDocumentObserver – интерфейс наблюдателя за документом

Методы

Метод	Описание
OpenFile	Открывает файл
WaitFile	Ожидает закрытие файла

OpenFile – открыть файл

Синтаксис:

```
procedure OpenFile(DeleteFileOnInitErrors: WordBool);
```

Параметр:

- *DeleteFileOnInitErrors* – признак возможности удаления файла наблюдателем самостоятельно, если при открытии документа возникли ошибки.

Описание:

Метод открывает файл для пользователя. В случае возникновения ошибок происходит исключение.

WaitFile – дождаться закрытия файла пользователем

Синтаксис:

```
procedure WaitFile;
```

Описание:

Метод следует вызывать только после вызова OpenFile.

Модуль расширения файлового хранилища

Для создания модуля расширения файлового хранилища необходимо реализовать интерфейсы IFileStoragePlugin и IShadowCopySupportPlugin, если модуль поддерживает работу с теневыми копиями.

IFileStoragePlugin – интерфейс модуля расширения файлового хранилища

Методы

Метод	Описание
InitStorage	Инициализирует новое хранилище
DeleteDocument	Удаляет документ
RestoreFromRecycleBin	Восстанавливает документ из корзины
RemoveFromRecycleBin	Безвозвратно удаляет документ из корзины
GetVersionsCount	Получает количество версий документа
CheckOut	Извлекает документ из хранилища
CreateNew	Создает новый файл документа в хранилище
CheckIn	Возвращает документ в хранилище
DeleteDocumentVersion	Удаляет версию документа
CopyDocumentVersion	Копирует версию документа
GetVersionSize	Получает размер версии документа
GetContents	Получает содержимое хранилища
UpdateMetadata	Обновляет метаданные документа

InitStorage – инициализировать новое хранилище

Синтаксис:

```
procedure InitStorage(
  const AccessInfo: IAccessInfo;
  const StorageInfo: IStorageInfo);
```

Параметры:

- *AccessInfo* – права доступа текущего пользователя к файлам хранилища;
- *StorageInfo* – параметры хранилища.

Описание:

Метод инициализирует файловое хранилище. Если файловое хранилище еще не создано, то метод создает его.

DeleteDocument – удалить документ

Синтаксис:

```
procedure DeleteDocument(
  const AccessInfo: IAccessInfo;
  const StorageInfo: IStorageInfo;
  DocumentID: Integer;
  ToRecycleBin: WordBool);
```

Параметры:

- *AccessInfo* – права доступа текущего пользователя к файлам хранилища;
- *StorageInfo* – параметры хранилища;
- *DocumentID* – ИД документа;
- *ToRecycleBin* – признак удаления документа в корзину: **True**, если документ удаляется в корзину, **False**, если документ удаляется безвозвратно. Документ можно восстановить из корзины.

Описание:

Метод удаляет документ с идентификатором *DocumentID*.

RestoreFromRecycleBin – восстановить документ из корзины

Синтаксис:

```
procedure RestoreFromRecycleBin(
  const AccessInfo: IAccessInfo;
  const StorageInfo: IStorageInfo;
  DocumentID: Integer);
```

Параметры:

- *AccessInfo* – права доступа текущего пользователя к файлам хранилища;
- *StorageInfo* – параметры хранилища;
- *DocumentID* – ИД документа.

Описание:

Метод восстанавливает из корзины удалённый документ в хранилище.

RemoveFromRecycleBin – удалить документ из корзины безвозвратно

Синтаксис:

```
procedure RemoveFromRecycleBin(
  const AccessInfo: IAccessInfo;
  const StorageInfo: IStorageInfo;
  DocumentID: Integer;
  NeedCheckIn: WordBool);
```

Параметры:

- *AccessInfo* – права доступа текущего пользователя к файлам хранилища;
- *StorageInfo* – параметры хранилища;
- *DocumentID* – ИД документа;
- *NeedCheckIn* – признак ~~необходимости~~ фиксирования удаления: **True**, если удаление документа фиксируется в базе данных внешней системы, иначе **False**. Если нет уверенности в том, что необходимо фиксировать удаление документов, установите значение **False**.

Описание:

Метод безвозвратно удаляет из корзины ранее удалённый документ.

GetVersionsCount – получить количество версий документа

Синтаксис:

```
function GetVersionsCount(
  const AccessInfo: IAccessInfo;
  const StorageInfo: IStorageInfo;
  DocumentID: Integer): Integer;
```

Параметры:

- *AccessInfo* – права доступа текущего пользователя к файлам хранилища;
- *StorageInfo* – параметры хранилища;
- *DocumentID* – ИД документа.

Возвращаемое значение:

Количество версий документа.

Описание:

Метод возвращает количество версий документа с идентификатором *DocumentID*. Если документа не существует, то возвращается значение **-1**.

CheckOut – извлечь документ из хранилища

Синтаксис:

```
function CheckOut(
  const AccessInfo: IAccessInfo;
  const StorageInfo: IStorageInfo;
  DocumentID: Integer;
  VersionNumber: Integer;
  const DocumentName: WideString;
  const DocumentExt: WideString;
  DesiredAccessRights: TDeaAccessRights): WideString;
```

Параметры:

- *AccessInfo* – права доступа текущего пользователя к файлам хранилища;
- *StorageInfo* – параметры хранилища;
- *DocumentID* – ИД документа;
- *VersionNumber* – номер версии документа;

- *DocumentName* – наименование документа;
- *DocumentExt* – расширение файла документа;
- *DesiredAccessRights* – тип прав, с которыми пользователь выполняет действия над документом после извлечения его из хранилища.

Возвращаемое значение:

Полный путь к файлу документа.

Описание:

Метод извлекает из хранилища указанную версию документа с идентификатором *DocumentID* и сохраняет её в файл с расширением *DocumentExt*.

CreateNew – создать в хранилище файл документа

Синтаксис:

```
function CreateNew(
  const AccessInfo: IAccessInfo;
  const StorageInfo: IStorageInfo;
  DocumentID: Integer;
  VersionNumber: Integer;
  const DocumentName: WideString;
  const DocumentExt: WideString;
  DesiredAccessRights: TDeaAccessRights): WideString;
```

Параметры:

- *AccessInfo* – права доступа текущего пользователя к файлам хранилища;
- *StorageInfo* – параметры хранилища;
- *DocumentID* – ИД документа;
- *VersionNumber* – номер версии документа;
- *DocumentName* – наименование документа;
- *DocumentExt* – расширение файла документа;
- *DesiredAccessRights* – тип прав, с которыми пользователь выполняет действия над документом после его создания.

Возвращаемое значение:

Полный путь к файлу созданного документа.

Описание:

Метод создаёт в хранилище пустой файл документа с указанными параметрами.

CheckIn – вернуть документ в хранилище

Синтаксис:

```
procedure CheckIn(
  const AccessInfo: IAccessInfo;
  const StorageInfo: IStorageInfo;
  DocumentID: Integer;
  VersionNumber: Integer;
  const FileName: WideString);
```

Параметры:

- *AccessInfo* – права доступа текущего пользователя к файлам хранилища;
- *StorageInfo* – параметры хранилища;
- *DocumentID* – ИД документа;
- *VersionNumber* – номер версии документа;
- *FileName* – имя файла документа.

Описание:

Метод возвращает в хранилище файл, извлеченный с помощью метода CheckOut.

DeleteDocumentVersion – удалить версию документа

Синтаксис:

```
procedure DeleteDocumentVersion(
  const AccessInfo: IAccessInfo;
  const StorageInfo: IStorageInfo;
  DocumentID: Integer;
  VersionNumber: Integer);
```

Параметры:

- *AccessInfo* – права доступа текущего пользователя к файлам хранилища;
- *StorageInfo* – параметры хранилища;
- *DocumentID* – ИД документа;
- *VersionNumber* – номер версии документа.

Описание:

Метод удаляет указанную версию документа.

CopyDocumentVersion – копировать версию документа

Синтаксис:

```
procedure CopyDocumentVersion(
  const AccessInfo: IAccessInfo;
  const StorageInfo: IStorageInfo;
  SourceDocumentID: Integer;
  SourceVersionNumber: Integer;
  DestinationDocumentID: Integer;
  DestinationVersionNumber: Integer;
  const DestinationDocumentName: WideString;
  const DestinationDocumentExt: WideString);
```

Параметры:

- *AccessInfo* – права доступа текущего пользователя к файлам хранилища;
- *StorageInfo* – параметры хранилища;
- *SourceDocumentID* – ИД документа-источника;
- *SourceVersionNumber* – номер версии документа-источника;
- *DestinationDocumentID* – ИД документа-приемника;
- *DestinationVersionNumber* – номер версии документа-приемника;

- *DestinationDocumentName* – наименование документа-приемника;
- *DestinationDocumentExt* – расширение документа-приемника.

Описание:

Копирует указанную версию документа-источника с идентификатором *SourceDocumentID* в указанную версию документа-приемника с идентификатором *DestinationDocumentID*. Наименование и расширение документа-приемника может не совпадать с наименованием и расширением документа-источника, в этом случае необходимо указать новые значения параметров *DestinationDocumentName* и *DestinationDocumentExt*.

GetVersionSize – получить размер версии документа

Синтаксис:

```
function GetVersionSize(
  const AccessInfo: IAccessInfo;
  const StorageInfo: IStorageInfo;
  DocumentID: Integer;
  VersionNumber: Integer;
  const VersionExt: WideString): Int64;
```

Параметры:

- *AccessInfo* – права доступа текущего пользователя к файлам хранилища;
- *StorageInfo* – параметры хранилища;
- *DocumentID* – ИД документа;
- *VersionNumber* – номер версии документа;
- *VersionExt* – расширение версии документа. Если расширение неизвестно, то значение параметра – пустая строка.

Возвращаемое значение:

Размер файла указанной версии документа в байтах.

Описание:

Метод возвращает размер файла указанной версии документа в байтах. Если версия не найдена, метод возвращает значение **-1**.

GetContents – получить описание содержимого хранилища

Синтаксис:

```
function GetContents(
  const AccessInfo: IAccessInfo;
  const StorageInfo: IStorageInfo): IStorageContents;
```

Параметры:

- *AccessInfo* – права доступа текущего пользователя к файлам хранилища;
- *StorageInfo* – параметры хранилища.

Возвращаемое значение:

Описание содержимого хранилища. Метод возвращает объект IStorageContents.

Описание:

Метод создает и возвращает объект IStorageContents с описанием содержимого.

UpdateMetadata – обновить метаданные документа в хранилище

Синтаксис:

```
procedure UpdateMetadata(
  const AccessInfo: IAccessInfo;
  const StorageInfo: IStorageInfo;
  DocumentID: Integer;
  const XML: WideString);
```

Параметры:

- *AccessInfo* – права доступа текущего пользователя к файлам хранилища;
- *StorageInfo* – параметры хранилища;
- *DocumentID* – ИД документа;
- *XML* – метаданные документа. Если значение параметра – пустая строка, то метаданные документа в файловом хранилище не будут изменены.

Описание:

Метод привязывает метаданные указанного документа к метаданным из параметра *XML*. Метаданные формируются с помощью справочника **Метаданные структурированных объектов**.

После успешного изменения метаданных документа метод автоматически вызывается повторно с пустым значением параметра *XML* для подтверждения транзакции.

IShadowCopySupportPlugin – интерфейс модуля расширения файлового хранилища с поддержкой теневых копий

Методы

Метод	Описание
CreateDocumentVersionShadowCopy	Создает теневую копию версии документа
DeleteDocumentVersionShadowCopy	Удаляет теневую копию версии документа
DeleteDocumentVersionShadowCopies	Удаляет несколько теневых копий версии документа
CheckOutShadowCopy	Извлекает из хранилища теневую копию версии документа

CreateDocumentVersionShadowCopy – создать теневую копию версии документа

Синтаксис:

```
procedure CreateDocumentVersionShadowCopy(
  const AccessInfo: IAccessInfo;
  const StorageInfo: IStorageInfo;
  DocumentID: Integer;
```

```
  VersionNumber: Integer;
  const DocumentName: WideString;
  const VersionExt: WideString;
  HistoryRecordID: Integer);
```

Параметры:

- *AccessInfo* – права доступа текущего пользователя к файлам хранилища;
- *StorageInfo* – параметры хранилища;
- *DocumentID* – ИД документа;
- *VersionNumber* – номер версии документа;
- *DocumentName* – наименование документа;
- *VersionExt* – расширение файла документа;
- *HistoryRecordID* – номер теневой копии.

Описание:

Метод создает теневую копию версии документа. Номер теневой копии указывается в параметре *HistoryRecordID*.

DeleteDocumentVersionShadowCopy – удалить теневую копию версии документа

Синтаксис:

```
procedure DeleteDocumentVersionShadowCopy(
  const AccessInfo: IAccessInfo;
  const StorageInfo: IStorageInfo;
  DocumentID: Integer;
  VersionNumber: Integer;
  HistoryRecordID: Integer);
```

Параметры:

- *AccessInfo* – права доступа текущего пользователя к файлам хранилища;
- *StorageInfo* – параметры хранилища;
- *DocumentID* – ИД документа;
- *VersionNumber* – номер версии документа;
- *HistoryRecordID* – номер теневой копии.

Описание:

Метод удаляет теневую копию версии документа.

DeleteDocumentVersionShadowCopies – удалить несколько теневых копий версии документа

Синтаксис:

```
procedure DeleteDocumentVersionShadowCopies(
  const AccessInfo: IAccessInfo;
  const StorageInfo: IStorageInfo;
  DocumentID: Integer;
  VersionNumber: Integer;
```

```
  ShadowCopyCount: Integer);
```

Параметры:

- *AccessInfo* – права доступа текущего пользователя к файлам хранилища;
- *StorageInfo* – параметры хранилища;
- *DocumentID* – ИД документа;
- *VersionNumber* – номер версии документа;
- *ShadowCopyCount* – количество оставшихся теневых копий~~, которые должны остаться~~.

Описание:

Для версии документа с номером *VersionNumber* метод удаляет теневые копии с первой, пока количество оставшихся теневых копий не станет равно *ShadowCopyCount*.

CheckOutShadowCopy – извлечь из хранилища теневую копию документа

Синтаксис:

```
function CheckOutShadowCopy(
  const AccessInfo: IAccessInfo;
  const StorageInfo: IStorageInfo;
  DocumentID: Integer;
  VersionNumber: Integer;
  const DocumentName: WideString;
  const DocumentExt: WideString;
  HistoryRecordID: Integer): WideString;
```

Параметры:

- *AccessInfo* – права доступа текущего пользователя к файлам хранилища;
- *StorageInfo* – параметры хранилища;
- *DocumentID* – ИД документа;
- *VersionNumber* – номер версии документа;
- *DocumentName* – наименование документа;
- *DocumentExt* – расширение файла документа;
- *HistoryRecordID* – номер теневой копии.

Возвращаемое значение:

Полный путь к файлу теневой копии документа.

Описание:

Метод извлекает из хранилища указанную теневую копию версии документа и сохраняет ее на диске в файл с расширением *DocumentExt*.

IAccessInfo – вспомогательный интерфейс для получения информации о правах доступа

Свойства

Свойство	Описание
UserName	Имя пользователя

UserName – имя пользователя

Синтаксис:

```
property UserName: WideString read Get_UserName;
```

Описание:

Свойство возвращает имя пользователя, от имени которого выполняются методы интерфейса IFileStoragePlugin. При выполнении действий над документами происходит проверка прав этого пользователя.

IStorageInfo – вспомогательный интерфейс для получения параметров хранилища

Свойства

Свойство	Описание
Code	Код хранилища
RootFolder	Корневой каталог хранилища

Code – код хранилища

Синтаксис:

```
property Code: WideString read Get_Code;
```

Описание:

Свойство возвращает код хранилища.

RootFolder – корневой каталог хранилища

Синтаксис:

```
property RootFolder: WideString read Get_RootFolder;
```

Описание:

Свойство возвращает локальный путь к корневой папке файлового хранилища.

IStorageContents – вспомогательный интерфейс для доступа к содержимому хранилища

Методы

Метод	Описание
Add	Добавляет элемент в описание содержимого хранилища
Insert	Вставляет элемент в указанную позицию описания содержимого хранилища

Свойства

Свойство	Описание
Count	Количество элементов в описании содержимого хранилища
Item	Набор элементов описания содержимого хранилища

Add – добавить элемент в описание содержимого хранилища

Синтаксис:

```
procedure Add(Value: TStorageItem);
```

Параметры:

- *Value* – элемент описания содержимого хранилища.

Описание:

Метод добавляет элемент описания содержимого хранилища в конец описания. При этом значение свойства Count увеличивается на единицу.

Insert – вставить элемент в описание содержимого хранилища

Синтаксис:

```
procedure Insert(
  Index: Integer;
  Value: TStorageItem);
```

Параметры:

- *Index* – индекс элемента;
- *Value* – элемент описания содержимого хранилища.

Описание:

Метод добавляет элемент описания содержимого хранилища на позицию, указанную в параметре *Index*. В качестве параметра *Index* следует указывать значение от 0 до Count–1, в противном случае будет сгенерировано исключение. У всех элементов описания, следующих за добавляемым элементом, индекс увеличивается на 1. Соответственно увеличивается и значение свойства Count.

Count – количество элементов в описании содержимого хранилища

Синтаксис:

```
property Count: Integer read Get_Count;
```

Описание:

Свойство возвращает количество элементов в описании содержимого хранилища.

Item – элементы описания содержимого хранилища

Синтаксис:

```
property Item[Index: Integer]: TStorageItem read Get_Item;
```

Описание:

Свойство возвращает элемент описания содержимого хранилища по его индексу.