Сервер NOMAD 2.10. Инструкция по созданию плагинов

В документе описан порядок создания плагинов для расширения возможностей стандартного сервера NOMAD.

Основные понятия

NOMAD (Native Offline Mobile Applications for DIRECTUM)

Нативные мобильные приложения для DIRECTUM – DIRECTUM Jazz и DIRECTUM Solo. Предназначены для работы с документами, задачами и заданиями системы посредством мобильных устройств.

NOMAD-пользователь

Пользователь, работающий в клиентском приложении NOMAD. Например, пользователь приложения DIRECTUM Solo.

Клиентское приложение DIRECTUM Jazz (Jazz)

Мобильное приложение, предназначенное для работы с объектами системы DIRECTUM посредством мобильных устройств на базе операционных систем iOS и Android. Связь клиентского приложения с системой DIRECTUM осуществляется с помощью веб-сервиса NOMAD.

Клиентское приложение DIRECTUM Solo (Solo)

Мобильное приложение, предназначенное для работы с объектами системы DIRECTUM посредством планшета. Связь клиентского приложения с системой DIRECTUM осуществляется с помощью сервера NOMAD.

Сервер NOMAD

Набор серверных компонент, включающий в себя веб-сервис NOMAD, конфигуратор серверной части и прикладную разработку NOMAD. Предназначен для выполнения функций, запрашиваемых клиентским приложением, предоставления доступа к данным и т.п.

Веб-сервис NOMAD

Серверное приложение, предоставляющее методы авторизации, доступа к данным системы DIRECTUM.

Общие сведения

Разработка плагинов ведется в среде Microsoft Visual Studio.

Плагины могут использоваться только в предусмотренных точках расширения сервера NOMAD. Точка расширения представляет собой интерфейс программной платформы .NET. Методы интерфейса вызываются в коде сервера.

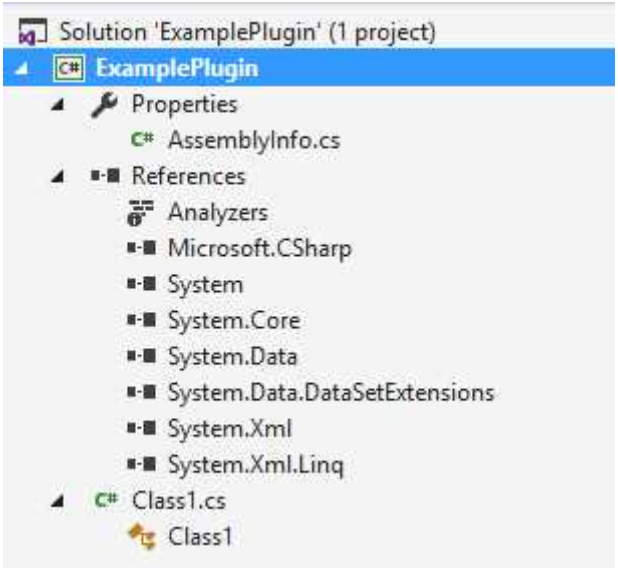
Хранение и загрузка плагинов настраиваются в файле Nomad.config.

Создание проекта в Microsoft Visual Studio

Создайте новый проект на основе шаблона Visual Studio:

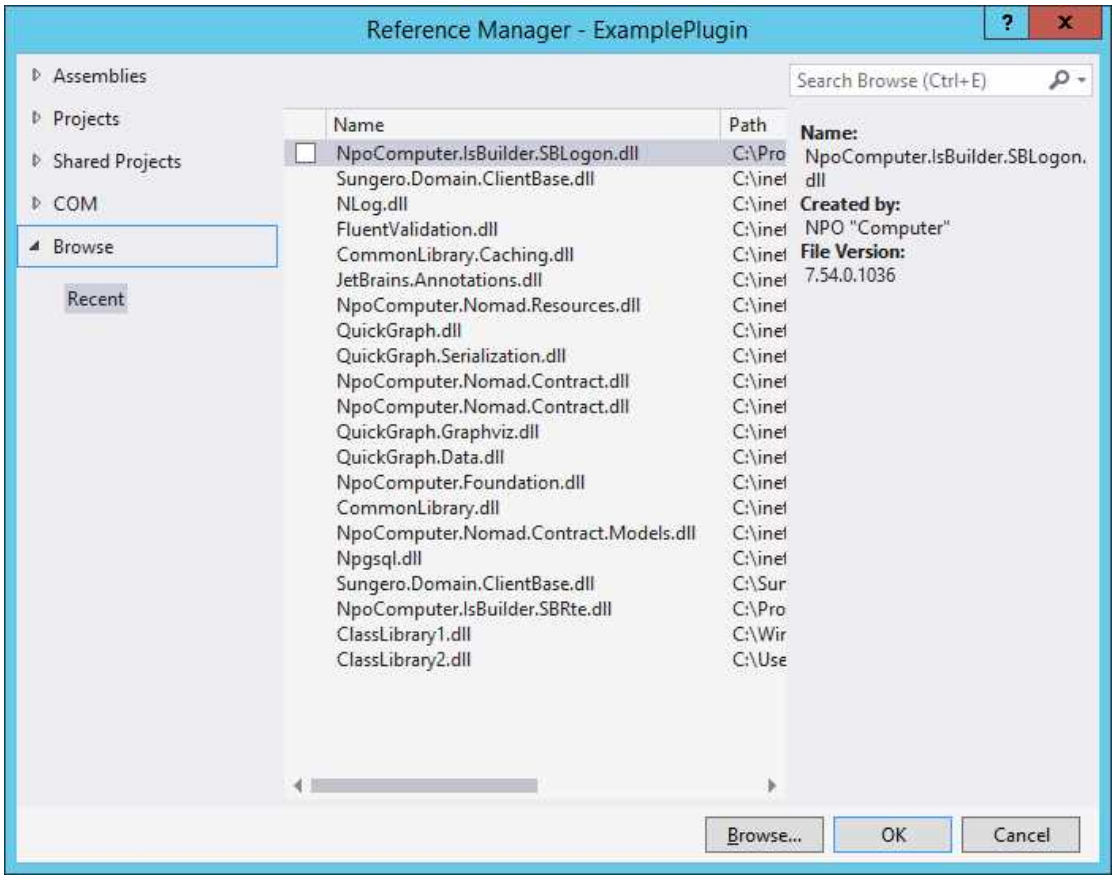
1. Запустите программу Microsoft Visual Studio.
2. На вкладке **File** последовательно выберите пункты **New**, **Project...**. Откроется окно «New Project».
3. В дереве настроек последовательно разверните вкладки:
   - Installed;
   - Templates;
   - Visual Basic – для разработки на языке Visual Basic;
   - Visual C# – для разработки на языке C#.
4. В области со списком шаблонов выберите пункт **Class Library (.NET Framework)**.
5. Укажите наименование, соответствующее назначению плагина и оканчивающееся на «Plugin».

   В результате будет создан новый проект на основе выбранного шаблона:



6. Созданный по умолчанию класс **Class1** переименуйте именем создаваемого плагина, т.к. он будет отображаться в настройках сервера NOMAD, и добавьте для него наследование от одного из интерфейсов плагинов.

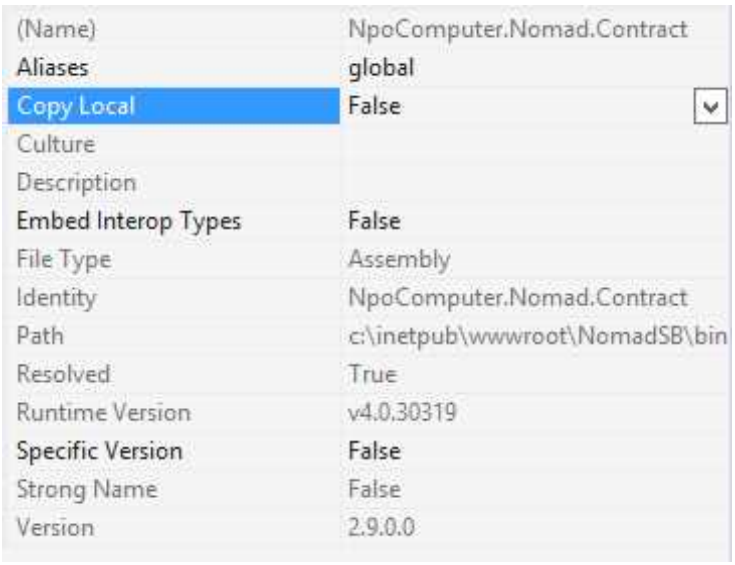
7. Для разрешения типа используемого интерфейса, а также используемых в нем типов, подключите к проекту соответствующие сборки, например NpoComputer.Nomad.Contract и NpoComputer.Nomad.Contract.Models. Для этого:
   a) В контекстном меню узла **References** выберите пункт **AddReference**.
   b) В открывшемся окне перейдите на вкладку «Browse»:



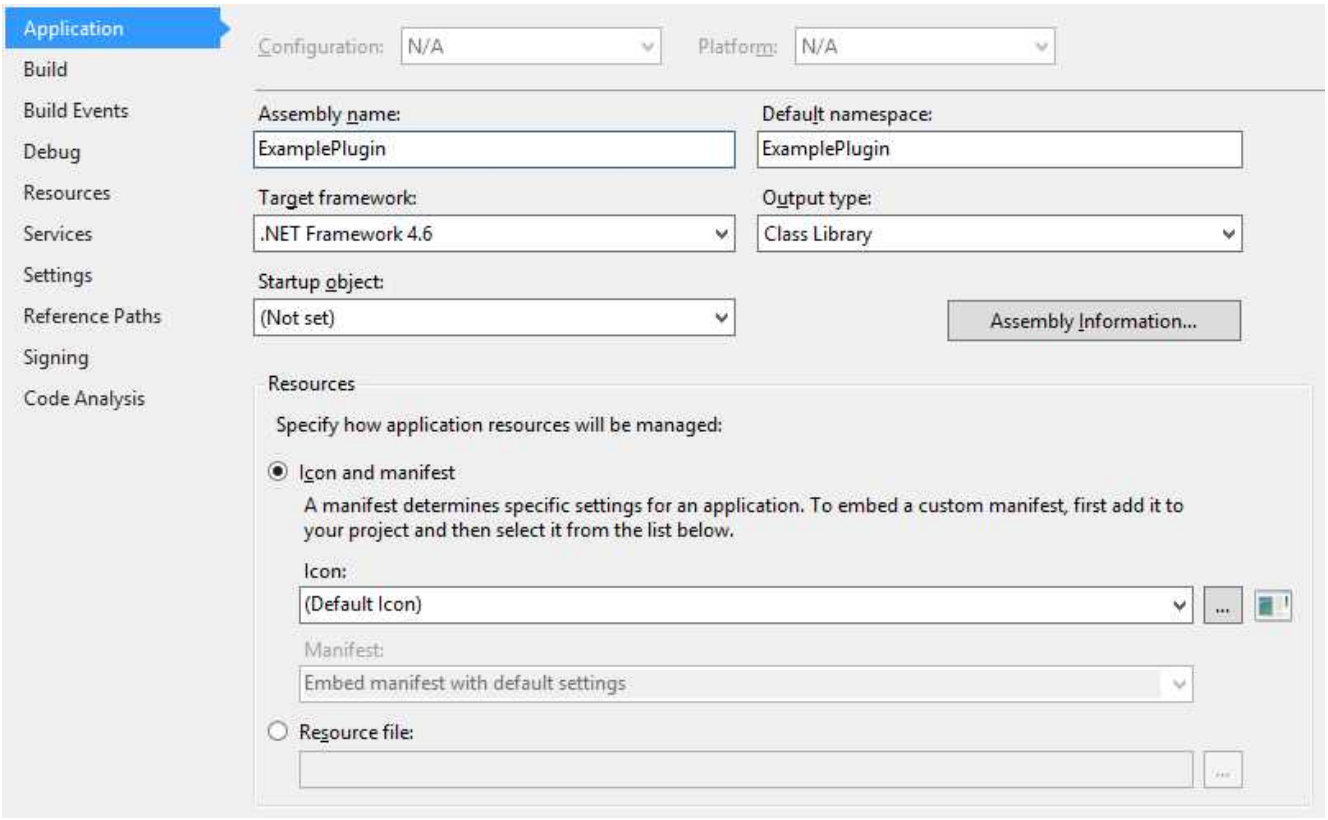
   c) Нажмите на кнопку **Browse...**. Откроется диалог выбора файлов.
   d) Выберите необходимую сборку из каталога **bin** установленного сервера NOMAD. Каталог сервера NOMAD может быть перенесен на текущий компьютер копированием с рабочего сервера или предварительно развернут на текущей компьютере.

   При успешной загрузке сборка будет отображаться в дереве проекта в узле **References**.
8. В контекстного меню сборки выберите пункт **Properties**.

9. В параметре **Copy Local** установите значение **False**:



10. В контекстном меню проекта выберите пункт **Properties**.
11. Перейдите на вкладку «Application» и в поле **Target framework** у становите значение **.NET Framework 4.6**. Эта версия соответствует версии подключаемых сборок сервера NOMAD:



12. Реализуйте методы выбранного интерфейса.

Интерфейсы

Для написания плагинов используются интерфейсы:

- IAuthenticationExtension – интерфейс плагина для аутентификации;
- IDeviceValidationPlugin – интерфейс плагина проверки устройств, с которых выполняется вход;
- ISoapTransformPlugin – интерфейс плагина преобразования SOAP-содержимого запросов и ответов сервера NOMAD;
- IContainerReaderPlugin - интерфейс плагина для работы с файлами, поддерживающими распаковку, например, архивы, письма с вложениями и т.д.;
- IJobExtension – интерфейс плагина для работы с заданиями.

Все используемые интерфейсы являются дочерними к интерфейсу IPlugin.

Описание плагинов, реализованных в данных интерфейсах и входящих в состав сервера NOMAD, см. в документе «Сервер NOMAD. Инструкция по установке и настройке», входит в комплект поставки.

IPlugin – родительский интерфейс

Не используется в качестве самостоятельной точки расширения. Предоставляет метод Initialize – инициализация плагина.

Метод Initialize – инициализация плагина

Синтаксис:

```
void Initialize();
```

Описание:

Метод вызывается при запуске сервера NOMAD.

IAuthenticationExtension – интерфейс плагина аутентификации пользователей

Позволяет вести учет входа пользователей. Предоставляет методы:

- ValidateBeforeLogin – проверить пользователя по реквизитам аутентификации;
- ValidateAfterLogin – проверить пользователя после входа по текущему контексту.

Метод ValidateBeforeLogin – поверить пользователя по реквизитам аутентификации

Синтаксис:

```
void ValidateBeforeLogin(AuthenticationCredentials credentials);
```

Параметры:

- *credentials* – реквизиты аутентификации в виде объекта класса AuthenticationCredentials.

Описание:

Метод вызывается при аутентификации пользователя до входа в систему. В методе могут выполняться дополнительные проверки по реквизитам аутентификации, например, разрешенному или запрещенному списку логинов.

Если проверка не будет выполнена, метод генерирует исключение LoginFailedException.

Класс AuthenticationCredentials – реквизиты аутентификации

Свойства:

Свойство	Тип	Описание
AuthenticationType	Элемент перечисления AuthenticationType	Тип аутентификации
LoginName	Строка	Имя учетной записи
Password	Объект класса **SecureString**	Пароль
Verify	Неизвестный тип	Проверяет полноту указанных данных

Класс AuthenticationType – тип аутентификации

Член перечисления	Значение	Описание
ClearText	0	Аутентификация по паролю
WindowsPassThrough	1	Сквозная Windows-аутентификация

Метод ValidateAfterLogin – проверить пользователя после входа по текущему контексту

Синтаксис:

```
void ValidateAfterLogin(IContext context);
```

Параметры:

- *context* – контекст взаимодействия с системой от имени пользователя в виде объекта IContext.

Описание:

Метод вызывается при аутентификации пользователя после входа в систему. В методе могут выполняться проверки по данным пользователя, полученным из его контекста взаимодействия с системой. Например, проверки по вхождению в группы.

Если проверка не будет выполнена, метод генерирует исключение LoginFailedException.

Класс IContext – контекст взаимодействия с СЭД от имени пользователя

Позволяет создать сессию, посредством которой возможна работа с объектами системой:

```
ISession CreateSession();
```

ISession определяет методы работы с объектами СЭД. Содержит методы создания и получения объектов СЭД.

Исключение LoginFailedException – ошибка аутентификации пользователя

Синтаксис:
```
public LoginFailedException(bool showUser, string message);
```

Параметры:

- *showUser* – признак отображения сообщения конечному пользователю;
- *message* – текстовое сообщение.

IDeviceValidationPlugin – интерфейс плагина проверки устройств

Предоставляет метод ValidateDevice – выполнить проверку устройства.

Метод ValidateDevice – выполнить проверку устройства

Синтаксис:
```
void ValidateDevice(IDeviceInfo deviceInfo, IUserInfo userInfo);
```

Параметры:

- *deviceInfo* – информация об устройстве, с которого выполняется подключение в виде объекта класса IDeviceInfo;
- *userInfo* – информация о пользователе в виде объекта класса IUserInfo.

Описание:

Метод вызывается при входе пользователя с устройства, не отмеченного как разрешенное.

На основе информации об устройстве и пользователе метод позволяет разрешить или запретить вход с устройства.

Дополнительно можно реализовать проверку по второму фактору защиты, например, через отправку письма на электронную почту пользователя с описанием действий для разрешения или запрета входа с устройства.

Разрешение или запрет входа с устройства осуществляется с помощью типа IDeviceManager.

Если вход пользователя с устройства не осуществляется, метод генерирует исключение NomadException.

В результате проверки возможны варианты:

- устройство отмечено как разрешенное – будет выполнен вход с устройства;
- устройство отмечено как запрещенное – сообщение исключения будет записано в лог-файл, и клиентскому приложению будет отправлена команда удаления данных из приложения;
- не удалось выполнить проверку или она еще не завешена – в клиентском приложении будет отображено сообщение с причиной неудачного входа.

Класс IDeviceInfo – информация об устройстве

Свойства:

Свойство	Тип	Описание
AppVersion	Строка	Версия приложения
Id	Целое число	Идентификатор устройства в сервере NOMAD
Name	Строка	Имя устройства, модель
OSVersion	Строка	Версия ОС

Класс IUserInfo – информация о пользователе

Свойства:

Свойство	Тип	Описание
Email	Строка	Адрес электронной почты
Id	Целое число	ИД пользователя
IsAdmin	Логический	Признак того, что пользователь входит в группу администраторов
Login	Строка	Логин пользователя
Name	Строка	Полное имя пользователя

Класс IDeviceManager – разрешение или запрет входа с устройства

Класс может быть получен с помощью вызова:

```
var deviceManager = Dependency.Resolve<IDeviceManager>();
```

Класс предоставляет методы для разрешения или запрета входа с устройства по его идентификатору:

- **EnableDevice** – пометить устройство с идентификатором *deviceId* как разрешенное.

  Синтаксис:

  void EnableDevice(int deviceId);

- **DisableDevice** – пометить устройство с идентификатором *deviceId* как запрещенное. При следующей попытке входа на устройство будет отправлена команда удаления данных приложения.

  Синтаксис:

  void DisableDevice(int deviceId);

- **GetDeviceInfo** – получить информацию об устройстве с идентификатором *deviceId*.

  Синтаксис:

  IDeviceInfo GetDeviceInfo(int deviceId);

Устройства, используемые пользователем, хранятся в таблице базы данных **NOMADUserDevices**.

Исключение NomadException – ошибка входа пользователя с устройства

Синтаксис:

```
public NomadException(int errorCode, ErrorOption option, string message);
```

Параметры:

- *errorCode* – код ошибки:
- *option* – параметры ошибки;
- *message* – текстовое сообщение.

ISoapTransformPlugin – интерфейс плагина преобразования SOAP-содержимого запросов и ответов сервера NOMAD

Позволяет производить преобразование строкового представления сообщений, передаваемых между сервером NOMAD и клиентскими приложениями.

Предоставляем методы:

- TransformInputMessage – преобразовать SOAP-содержимое запросов к серверу;
- TransformOutputMessage – преобразовать SOAP-содержимое ответов сервера.

Метод TransformInputMessage – преобразовать SOAP-содержимое запросов к серверу

Синтаксис:

```
string TransformInputMessage(string message);
```

Параметры:

- *message* – SOAP-сообщение запроса серверу NOMAD от клиентского приложения.

Возвращаемое значение:

Строка, полученная путем преобразования SOAP-содержимого запроса к серверу.

Описание:

Метод вызывается при получении запроса от клиентского приложения перед его обработкой.

Метод TransformOutputMessage – преобразовать SOAP-содержимое ответов сервера

Синтаксис:

```
string TransformOutputMessage (string message);
```

Параметры:

- *message* – SOAP-сообщение ответа сервера NOMAD клиентскому приложению.

Возвращаемое значение:

Строка, полученная путем преобразования SOAP-содержимого ответа сервера.

Описание:
Метод вызывается после формирования ответа клиентскому приложению перед его непосредственной передачей.

IContainerReaderPlugin – интерфейс плагина работы с файлами, поддерживающими распаковку

Предоставляет методы:

- GetContainer – получить файловый контейнер;
- GetSupportedExtensions – получить список расширений документов, которые может обрабатывать плагин.

Метод GetContainer – получить файловый контейнер

Синтаксис:

```
IContainer GetContainer(Stream stream);
```

Параметры:

- *stream* – исходный документ в виде потока.

Возвращаемое значение:

Файловый контейнер в виде интерфейса IContainer.

Метод GetSupportedExtensions – получить список расширений документов, которые может обрабатывать плагин

Синтаксис:

```
IEnumerable<string> GetSupportedExtensions();
```

Возвращаемое значение:

Список расширений, которые может обрабатывать плагин, в виде строк.

IContainer – интерфейс, представляющий отдельный контейнер

Предоставляет методы:

- GetEntries – получить объекты верхнего уровня;
- GetChildEntries – получить дочерние объекты текущего объекта;
- GetEntryStream – получить содержимое объекта.

GetEntries – получить объекты верхнего уровня

Синтаксис:

```
IEnumerable<IContainerEntry> GetEntries();
```

Возвращаемое значение:

Объекты верхнего уровня в виде списка объектов класса IContainerEntry.

GetChildEntries – получить дочерние объекты текущего объекта

Синтаксис:

```
IEnumerable<IContainerEntry> GetChildEntries(string entryKey);
```

Параметры:

- *entryKey* – ключ текущего объекта в контейнере.

Возвращаемое значение:

Дочерние объекты в виде списка объектов класса IContainerEntry.

GetEntryStream – получить содержимое объекта

Синтаксис:
```
Stream GetEntryStream(string entryKey);
```

Параметры:

- *entryKey* – ключ объекта в контейнере.

Возвращаемое значение:

Исходный документ в виде потока.

Класс IContainerEntry – информация об объекте контейнера

Свойства:

Свойство	Тип	Описание
Created	Дата	Дата создания объекта
IsDirectory	Логический	Признак того, что объект является каталогом
Key	Строка	Уникальный ключ объекта в контейнере
Modified	Дата	Дата изменения объекта
Name	Строка	Имя объекта, включая расширение
Size	64-разрядное целое число	Размер объекта

IJobExtension – интерфейс плагина работы с заданиями

Позволяет встраиваться в логику обработки запрашиваемых параметров.

Предоставляет метод GetAvailableParamsValues – получить допустимые значения запрашиваемых параметров задания.

Метод GetAvailableParamsValues – получить допустимые значения запрашиваемых параметров задания

Синтаксис:
```
IEnumerable<ParamValueModel> GetAvailableParamsValues(out bool handled, Context context, int jobId, string standardRouteCode, string askableParamCode, string likeReqValue, int offset, int count, out string warningMessage);
```

Параметры:

- *handled* – признак обработки;
- *context* – контекст пользователя;
- *jobId* – идентификатор задания;
- *standardRouteCode* – код типового маршрута;
- *askableParamCode* – код запрашиваемого параметра;
- *likeReqValue* – значение для like-поиска по реквизиту;

- *offset* – количество записей, которые нужно пропустить;
- *count* – количество записей, которые нужно вернуть;
- *warningMessage* – предупреждение, отображаемое пользователю.

Возвращаемое значение:

Перечисление моделей значений параметров в виде объектов класса ParamValueModel.

Описание:

Метод вызывается при запросе допустимых значений параметров варианта выполнения задания.

После обработки запроса метод меняет значение параметра *handled* на **true**.

Класс ParamValueModel – информация о значении параметра

Свойства:

Свойство	Тип	Описание
AuthorId	Целое число	ИД автора документа (для типа параметра **электронный документ**)
AuthorFullName	Строка	Полное имя автора документа (для типа параметра **электронный документ**)
Created	Дата	Дата создания документа (для типа параметра **электронный документ**)
Display	Строка	Значение, отображаемое пользователю
Extension	Строка	Расширения файла (для типа параметра **электронный документ**)
HaveSignature	Элемент перечисления **SignType**	Признак наличия подписей (для типа параметра **электронный документ**)
IsEncrypted	Логический	Признак зашифрованности (для типа параметра **электронный документ** или **объект делового процесса**)
Modified	Дата	Дата модификации документа (для типа параметра **электронный документ**)
Value	Строка	Значение параметра

Хранение и загрузка плагинов

Настройки хранения и загрузки плагинов задаются в файле Nomad.config в секции **plugins**.

По умолчанию плагины хранятся в папке сервиса App_Data\Plugins:

```
<plugins folder="~\App_Data\Plugins\">
```

При необходимости папку для хранения плагинов можно изменить. Например, когда сервер NOMAD используется в кластере и нужно указать одну общую папку с плагинами.

Чтобы подключить плагин, пропишите его в атрибуте **add**:

```
<add type="<Наименование сборки плагина>.<Наименование класса плагина>"/>
```