

Система «eDocLib»

Версия 2.6

Выпуск 1

Руководство прикладного программиста

«Электронные Офисные Системы»: 107113 Москва, ул. Шумкина, 20, стр. 1.

Тел/факс: (495) 221-24-31, E-mail: support@eos.ru

Документ разработан специалистами компании «Электронные
Офисные Системы».

Copyright © 1996-2016 «Электронные Офисные Системы». Все
права защищены. «eDocLib» является торговой маркой
предприятия «Электронные Офисные Системы».

Все упомянутые в данном издании товарные знаки и
зарегистрированные товарные знаки принадлежат своим
законным владельцам.

ОГЛАВЛЕНИЕ

EDOSCLIB 2.6: РУКОВОДСТВО ПРИКЛАДНОГО ПРОГРАММИСТА	1
ВВЕДЕНИЕ	7
Веб-сервис: гибкость в сочетании с удобством	7
МЕТАДАННЫЕ И КОНФИГУРАТОР	8
Использование сгенерированного кода.....	8
Внутреннее хранение данных и формат DataSet	10
Позднее связывание	10
Идентификаторы типов	11
ИНТЕРФЕЙС ISYSTEMAPI.....	11
Создание проекта в Visual Studio.....	11
Создание объекта интерфейса. Буферизованный и потоковый режимы.	11
Демонстрационный пример	13
Работа с пользователями и подключениями.....	13
Connect.....	14
ConnectAsWindowsUser.....	14
Disconnect	14
GetCurrentUser.....	14
SaveRecentDocument.....	15
Unlock	15
SaveSessionData.....	15
LoadSessionData	16
DeleteSessionData	16
GetAuthenticationMode	16
IsDisconnected.....	16
GetPrivateFoldersTree.....	16
GetMyACLForFolder	17
Работа со справочниками	18
LoadCatalog.....	18
LoadCatalogForUpdate.....	19
SaveCatalog	20
SearchCatalog	20
ExportTypeSchema	21
GetCatalogList.....	22
LoadCatalogRule	22
Unlock	23
GetMyACLForCatalog	24
ReplaceCatalogReferences	24
CreateCatalog	24
GetPrivateFoldersTree.....	25
GetMyACLForFolder	25
Работа с документами.....	25
CreateDocumentByGroup	25
CreateDocumentByGroupAndOther	26

CreateNewDocumentVersion	26
ReleaseNewDocument.....	27
GetRegistrationDocumentGroups	27
LoadDocument	28
LoadDocumentForUpdate	29
SaveDocument.....	29
SearchDocsByCriteries	30
LoadDocsByCriteries	31
GetMyACLForDocument.....	33
GetDocumentLog	33
LoadDocFile	33
GetPrivateSearches	34
RunPrivateSearch	34
AddDocumentsToFolders.....	35
MoveDocumentsFromFolderToFolders	35
RemoveDocumentsFromFolders	36
LockRecordsForUpdate	36
UnlockRecordsForUpdate	37
LockRecordsForInsertChild	37
UnlockRecordsForInsertChild	38
GetDocumentPrintData.....	39
SaveDocFile.....	39
Работа с оповещениями.....	40
CreateEventSubscription	40
RemoveEventSubscription	40
ConfirmEventNotification	40
SendEventNotification.....	40
Работа с подписью.....	41
GetDocumentPassport	41
CreateDocumentByGroupWithPassport	41
SEVCreatePassportFromDocId	41
SEVCreateDocFromPassport	41
SEVCreateDocFromPassportUnsaved.....	42
Работа с импортом и экспортом	42
ExportObject.....	42
ImportObject.....	42
Работа с метаданными.....	42
LoadActualMetadata.....	42
Работа с пользовательским хранилищем	43
SaveToCustomStorage	43
LoadFromCustomStorage.....	43
Работа с настройками системы.....	43
LoadSystemSettings	43
SaveSystemSettings	43
ПРИЛОЖЕНИЕ 1. СТАНДАРТНЫЕ ТИПЫ СИСТЕМЫ.....	44
Имена полей и секций	44
Базовые свойства объектов	45
Eos.DP.Wrapper.Base (Базовый объект)	45
Eos.DP.Wrapper.Head (Головная секция)	45
Линейные справочники системы.....	47
Eos.DP.Wrapper.Catalog (Справочник).....	47
Eos.DP.Wrapper.IdentityType (Вид удостоверения личности)	47

Eos.DP.Wrapper.AddressType (Тип адреса).....	47
Eos.DP.Wrapper.LinkType (Тип ссылки).....	47
Eos.DP.Wrapper.Numerator (Нумератор).....	47
Eos.DP.Wrapper.MessageType (Тип сообщения).....	47
Eos.DP.Wrapper.ColorFilters (Цветовые фильтры).....	48
Eos.DP.Wrapper.DocumentStatus (Статус документа).....	48
Eos.DP.Wrapper.ReportForm (Печатная форма).....	48
Eos.DP.Wrapper.PrintFormFormat (Формат печатной формы).....	48
Eos.DP.Wrapper.v2.5.1.DateRange (Диапазоны дат).....	49
Eos.DP.Wrapper.TaskTypeProcessing (Тип обработки поручения).....	49
Eos.DP.Wrapper.BarCodeType (Тип штрихкода).....	49
Eos.DP.Wrapper.WayType (Тип маршрута).....	49
Eos.DP.Wrapper.Docflow (Маршрут документа).....	49
Eos.DP.Wrapper.WorkflowType (Тип процесса).....	49
Eos.DP.Wrapper.WayStageType (Тип этапа маршрута).....	50
Eos.DP.Wrapper.VarType (Тип переменной).....	50
Eos.DP.Wrapper.FolderSMS (Папка СМР).....	50
Справочники с рубрикацией.....	51
Eos.DP.Wrapper.RubricatedCatalog (Справочник с рубрикацией).....	51
Eos.DP.Wrapper.Country (Страна).....	51
Eos.DP.Wrapper.DocumentGroup (Группа документов).....	51
Eos.DP.Wrapper.DocumentSearchSettings (Настройки поиска документов).....	54
Eos.DP.Wrapper.Folder (личная папка).....	55
Справочники "Контакт".....	56
Eos.DP.Wrapper.Uol.UolBase (Контакт).....	57
Eos.DP.Wrapper.Uol.Citizen(Гражданин).....	57
Eos.DP.Wrapper.Uol.Organization (Организация).....	58
Eos.DP.Wrapper.Uol.Official (Должностное лицо).....	59
Eos.DP.Wrapper.Uol.Department (Подразделение).....	59
Документы.....	60
Eos.DP.Wrapper.Doc.BaseDocument (Документ).....	60
Eos.DP.Wrapper.Doc.Task (Поручение).....	64
Eos.DP.Wrapper.Doc.Report (Отчёт).....	65
Eos.DP.Wrapper.Doc.OfficialDocument (Служебный документ) и его наследники.....	66

ПРИЛОЖЕНИЕ 2. СИНТАКСИС ПОИСКОВЫХ ВЫРАЖЕНИЙ МЕТОДА SEARCHCATALOG 68

Примеры.....	69
---------------------	-----------

ПРИЛОЖЕНИЕ 3. ЗАДАНИЕ КРИТЕРИЕВ ПОИСКА ДОКУМЕНТОВ. ОБЪЕКТ CRITERYGROUP 70

CrteryGroup.....	70
AddFieldCrtery.....	70
AddFullTextCrtery.....	71
AddGroupCrtery.....	71
AddNullCrtery.....	71
AddNotNullCrtery.....	72
AddParamCrtery.....	72
AddTextCrtery.....	73
AddSubqueryCrteryExists.....	73
AddSubqueryCrteryFrom.....	73
AddSubqueryCrteryIn.....	73

ПРИЛОЖЕНИЕ 4. СКРИПТЫ. 74

Скрипты значений по умолчанию	74
Скрипты копирования	75
Скрипты цветовых фильтров	76
Скрипты на сохранение документа	77
Catalogs	77
ПРИЛОЖЕНИЕ 5. ПЕЧАТНЫЕ ФОРМЫ.	79
Печатная форма – что это?	79
Что печатаем	79
Как печатаем	79
Создание печатной формы для документа	79
Выбор типа документов	79
Редактирование API-имен реквизитов в типе документов.	80
Получение схемы источника данных в XML формате.....	81
Создание печатной формы.....	81
Печать документа	82
Создание печатной формы поискового запроса	83
Получение схемы источника данных в XML формате для поискового запроса	83
Создание печатной формы.....	83
Печать результатов поиска	83
Импорт/экспорт печатных форм	85
Импорт печатной формы в систему	85
Экспорт печатной формы из системы.....	86
ПРИЛОЖЕНИЕ 6. ТРИГГЕРЫ.	87
Регистрация триггера	87
Какие сборки нужно включить в проект	88
Как из триггера работать с буфером документа.....	88
Команды, для которых можно создавать триггеры.....	89
ILoadDocumentCommand	89
LoadDocumentCommandEventArgs	89
ISaveDocumentCommand	90
SaveDocumentCommandEventArgs	90
ICreateDocumentByGroupCommand	90
CreateDocumentCommandEventArgs	91
Команды, которые можно вызывать из триггера.....	91
SearchCatalog	92
SearchCatalogCommandParameters	92
LoadCatalog	92
LoadCatalogCommandParameters.....	93
SaveCatalog	93
SaveCatalogCommandParameters	93
Примеры	94

Введение

Документальный сервер eDocLib оснащён интерфейсом прикладного программирования (API), позволяющим интегрировать продукт с различными информационными системами и существенно расширять функциональность системы eDocLib.

Данный документ описывает программные возможности API eDocLib и предполагает знание подсистемы «Конфигуратор» и знакомство с инфраструктурой System.Data в Microsoft.NET. Информацию по конфигуратору можно найти в Руководстве пользователя.

Веб–сервис: гибкость в сочетании с удобством

API системы eDocLib опубликован в виде веб-сервиса. Данное решение обладает следующими преимуществами:

- позволяет использовать для программирования любую программную среду, поддерживающую вызов веб-сервисов;
- обладает достаточной гибкостью, чтобы использовать не только стандартные, но и создаваемые пользователем типы данных (см. раздел «Метаданные и конфигуратор»);
- даёт возможность интегрировать eDocLib как с программными компонентами, находящимися в сети организации, так и с системами, доступными через Internet;
- Использует протокол HTTP, что позволяет работать даже в средах, где другие протоколы недоступны или заблокированы.

Веб-сервис системы публикуется автоматически после инсталляции веб-приложения и доступен всё время, пока работает Internet Information Services (IIS), не требуя развертывания дополнительных компонентов или программ. Для доступа к веб-сервису eDocLib используйте следующий адрес:

`http://servername/appname/systemapi.svc`

, где `servername` – сетевое имя компьютера, на котором установлена система eDocLib, `appname` – имя Web-приложения, выбранное на этапе установки системы.

Примечание: не забудьте, что этот адрес необходимо использовать при настройке программных компонентов. Если же вы введете его в окно браузера, вы просто увидите подсказку по работе с сервисом (Рис. 1)



Рис. 1. Подсказка по работе с сервисом SystemApi.

В данном руководстве будет объясняться наиболее удобный механизм использования API eDocLib: компонент Microsoft .NET 4.5 **Windows Communication Foundation (WCF)**. Будут приводиться примеры на языке C# 5.0, созданные в среде Visual Studio 2013.

Если же необходимо вызывать методы API из других систем, получите описание схемы API на языке WSDL (Web Services Description Language) и следуйте руководству по выбранной программной среде. Для получения WSDL используйте следующий адрес:

`http://servername/appname/systemapi.svc?wsdl`

Метаданные и конфигуратор

Система eDocLib позволяет пользователю, обладающему ролью «Конфигуратор», изменять структуры данных: создавать новые типы документов, справочники, добавлять и изменять их реквизиты, менять их системные параметры (имена таблиц и колонок базы данных и т.д.).

Информация, описывающая структуры данных (иначе говоря – *метаданные*) хранится в самой системе.

Использование сгенерированного кода

Система eDocLib генерирует вспомогательные классы для платформы .NET, которые отражают созданные в конфигураторе типы данных. Чтобы использовать эти классы, используйте ссылку «Экспортировать код» в интерфейсе конфигулятора (Рис. 2):

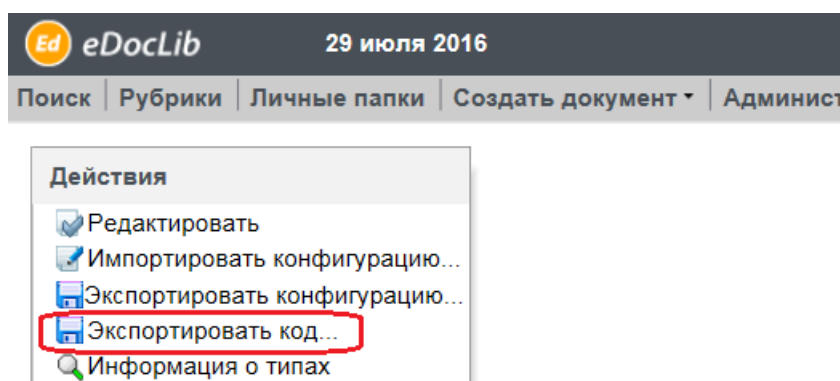


Рис.2. Экспорт сгенерированного кода.

Сохраните полученный файл **Types.dll** на диск и добавьте его в свой проект Visual Studio 2013.

Объекты (документы и справочники), добавленные в конфигураторе, превращаются в классы .NET. Поля объектов (простые и ссылочные реквизиты) – в свойства этих классов. Доступ к свойствам осуществляется через привычную нотацию:

```
// объект country - это запись справочника "Страна"

...

country.Code = 101;

country.Name = "Люксембург";
```

По умолчанию, создаваемым в конфигураторе типам и полям присваиваются уникальные имена, неудобные для запоминания человеком. Но вы можете контролировать именование объектов и полей, используя закладки "Тип" и "Элемент" в свойствах выбранных объектов (Рис. 3):

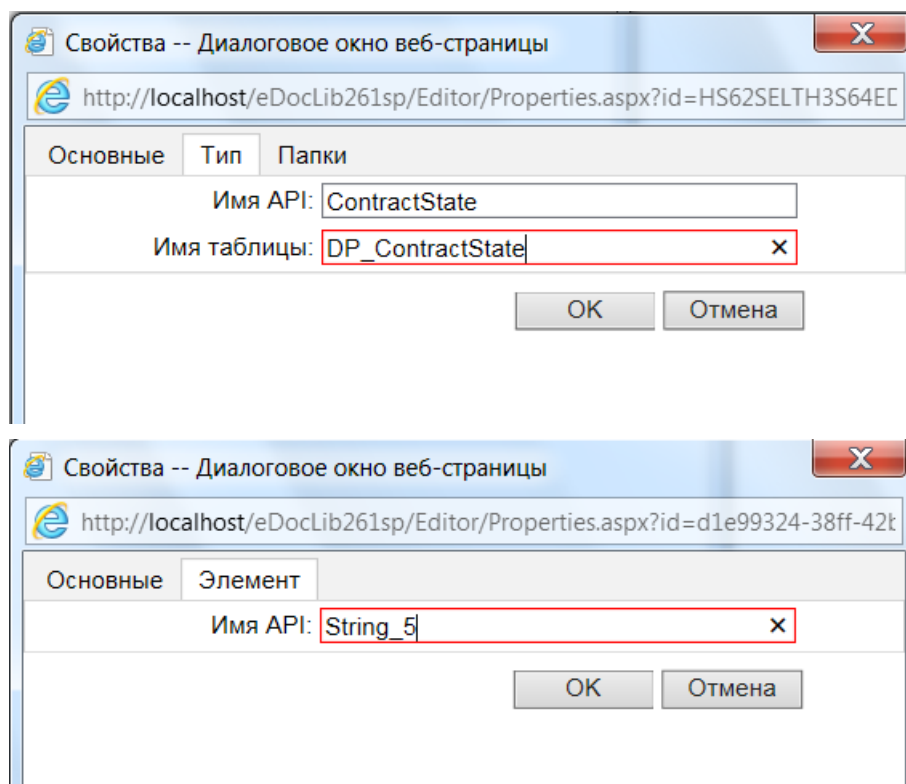


Рис. 3. Переименование имён API типов и элементов

После применения изменений вы можете экспортировать код, содержащий правильные имена API, и подключить его к проекту. Не забывайте, что любое изменение имён API может повлечь за собой негодность уже написанного кода, использующего старые имена.

Внутреннее хранение данных и формат DataSet

Методы API, однако, требуют не типов eDocLib, а объектов типа .NET DataSet, который предназначен для описания табличных структур данных. У сгенерированных типов есть встроенная поддержка DataSet. Фактически, они основаны на этом типе данных, и лишь «оборачивают» его в удобную для написания кода оболочку (отсюда название пространства имён системных объектов – “Wrappers” т.е., «обертки»).

При создании объекта «с нуля» необходимо вызвать метод `CreateBuffer(true)`, который создаст DataSet для хранения данных. Для того чтобы получить DataSet из существующего объекта, обратитесь к его свойству `Buffer`:

```
// Метод CreateOrganization - условный и приводится в демонстрационных целях  
  
var org = CreateOrganization();  
org.Name = "Филиал в Челябинске";  
org.Town = "Челябинск";  
api.SaveCatalog(sessionToken, org.Buffer);
```

(Методы интерфейса `ISystemApi` будут описаны далее).

Примечание: Помните, что система eDocLib полагается на формат DataSet при определении необходимых действий над объектом. Для выяснения, какие операции необходимо произвести для изменения, анализируются состояния строк DataSet-а. Например, для удаления записи справочника из БД загрузите эту запись, вызовите метод `Delete` у полученного объекта и сохраните изменённую запись. Не вызывайте метод `AcceptChanges` перед сохранением, иначе информация об изменениях строк будет утеряна, и система проигнорирует изменение данных.

Позднее связывание

Другой способ обращения к объектам позволяет абстрагироваться от конкретных типов. Вам достаточно знать, что в используемом документе или справочнике есть поля или секции, названные определенным образом. Ваш код, в результате, будет выполнять «позднее связывание», обращаясь по этим именам.

Эта возможность предоставляется благодаря формату .NET DataSet. Вот как может выглядеть предыдущий пример с использованием DataSet:

```
// Метод LoadOrganization, возможно, загружает объект из XML-файла  
  
DataSet org = LoadOrganization();  
org.Tables[0].Rows[0]["Name"] = "Филиал в Челябинске";  
org.Tables[0].Rows[0]["Town"] = "Челябинск";  
api.SaveCatalog(sessionToken, org);
```

Данный способ несколько сложнее в программировании, так как он предполагает знания особенностей объектов DataSet и не предоставляет помощи на этапе компиляции. Например, если вы совершите ошибку в имени поля, написав «Twn» вместо «Town», то вы не узнаете об этом до момента выполнения вашего кода.

Но благодаря позднему связыванию появляются возможности написания универсальных программ. Например, можно создать утилиту импорта справочников из другой системы посредством загрузки файлов формата XML. Достаточно убедиться, что файлы

сохраняются в нужном формате, и ваш код сможет выполнять импорт записей справочников любого типа (даже созданных после написания утилиты).

Идентификаторы типов

Многие методы API используют идентификаторы типов системы. Это строковые значения, уникально идентифицирующие тот или иной тип. При создании нового типа документа или справочника идентификатор типа генерируется автоматически.

Для получения значения идентификатора типов используйте конструкцию вида `class.TypeId`, где `class` – имя типа в сгенерированном коде.

Интерфейс `ISystemApi`

Создание проекта в Visual Studio

Для работы через API необходимо создать проект нужного типа (например, Windows Application) в Visual Studio. Добавьте в проект ссылку на Windows Communication Foundation (System.ServiceModel).

Для вызовов API предназначен интерфейс (.NET interface) `ISystemApi`. Для использования этого интерфейса подключите в проект сборку **Eos.DP.RemoteApi.dll**. Кроме того, потребуется включить сборку **Eos.DP.Core.dll**, описывающую работу с поисковыми критериями документов (она описана в Приложении 3).

Если планируется использовать сгенерированный Конфигуратором код вспомогательных классов, то включите в проект ещё две сборки: **Eos.DP.Wrapper.dll** и сгенерированную **Types.dll**. Если же используются только DataSet-ы, эти сборки не понадобятся.

Создание объекта интерфейса. Буферизованный и потоковый режимы.

Для создания подключения к серверу eDocLib можно использовать как декларативный способ (описание параметров подключения в файле App.Config), так и явную программную установку параметров соединения. Второй способ предпочтительнее, так как он позволяет легче контролировать настройки WCF. Документацию по декларативной настройке App.Config можно найти в MSDN (раздел <system.serviceModel>).

Ниже приведен пример обычной настройки параметров соединения для сервера "server1" и приложения "eDocLib":

```
public static ISystemApi CreateAPIClient()
{
    string address = "http://server1/edoclib/systemapi.svc";
    EndpointAddress endpoint = new EndpointAddress(address);
    System.ServiceModel.Channels.Binding binding;
    binding = CreateBufferedBinding();
    return ChannelFactory<ISystemApi>.CreateChannel(binding, endpoint);
}

private static System.ServiceModel.Channels.Binding CreateStreamedBinding()
{
    BasicHttpBinding binding = new BasicHttpBinding();
    binding.MessageEncoding = WSMessageEncoding.Mtom;

    // устанавливаем потоковый режим
    binding.TransferMode = TransferMode.Streamed;
}
```

```
binding.TextEncoding = Encoding.UTF8;
binding.MaxBufferPoolSize = 524288;
binding.MaxBufferSize = 65535; // размер буфера небольшой
binding.MaxReceivedMessageSize = 1000000000; // сообщения до 1 GB
binding.OpenTimeout = new TimeSpan(0, 2, 0);
binding.CloseTimeout = new TimeSpan(0, 2, 0);
binding.SendTimeout = new TimeSpan(0, 50, 0);
binding.ReceiveTimeout = new TimeSpan(0, 50, 0);
return binding;
}
```

Настроенная таким образом система поддерживает буферизованную передачу данных: размер любого сообщения (включая файлы документа) не может превышать ограничения на размер буфера (MaxBufferSize и MaxReceivedMessageSize, в байтах). При этом выделение значительных объемов памяти на сервере приложения может мешать работе других пользователей. Более того, при запросе слишком больших блоков данных вашему приложению может быть отказано в выделении памяти.

Если же требуется устойчивая работа с файлами большого размера (до 512 мегабайт), то система поддерживает так называемый потоковый режим. При этом передача данных осуществляется небольшими блоками, и отнимать память у других приложений не потребуется.

Для обеспечения потокового режима работы необходимо, чтобы при установке eDocLib была выключена опция "Использовать авторизацию Windows". (Windows-авторизация для потокового режима поддерживается в WCF только при соединении посредством HTTPS).

Ниже приведен пример настройки потового режима работы (авторизация по логину и паролю):

```
public static ISystemApi CreateAPIClient()
{
    string address = "http://server1/edoclib/systemapi.svc";
    EndpointAddress endpoint = new EndpointAddress(address);
    System.ServiceModel.Channels.Binding binding;
    binding = CreateStreamedBinding();
    return ChannelFactory<ISystemApi>.CreateChannel(binding, endpoint);
}

private static System.ServiceModel.Channels.Binding CreateStreamedBinding()
{
    BasicHttpBinding binding = new BasicHttpBinding();
    binding.MessageEncoding = WSMessageEncoding.Mtom;

    // устанавливаем потоковый режим
    binding.TransferMode = TransferMode.Streamed;
    binding.TextEncoding = Encoding.UTF8;
    binding.MaxBufferPoolSize = 524288;
    binding.MaxBufferSize = 65535; // размер буфера небольшой
    binding.MaxReceivedMessageSize = 1000000000; // сообщения до 1 GB
    binding.OpenTimeout = new TimeSpan(0, 2, 0);
    binding.CloseTimeout = new TimeSpan(0, 2, 0);
    binding.SendTimeout = new TimeSpan(0, 50, 0);
    binding.ReceiveTimeout = new TimeSpan(0, 50, 0);
    return binding;
}
```

Примечание: ваше приложение может совместить эти два метода, пытаясь создать сначала потоковое соединение, а если оно не поддерживается (при Windows-авторизации) – переключаться на буферизованный режим.

Демонстрационный пример

Чтобы получить более полное представление о программировании eDocLib, рекомендуется ознакомиться с демонстрационным примером **WCFClient**, поставляемым в составе данного руководства. Он содержит готовый проект для среды Visual Studio 2013, демонстрирующий создание Windows-приложения с использованием методов API: создания документов, добавления файлов, чтения документов и файлов, поиска документов (Рис. 4).

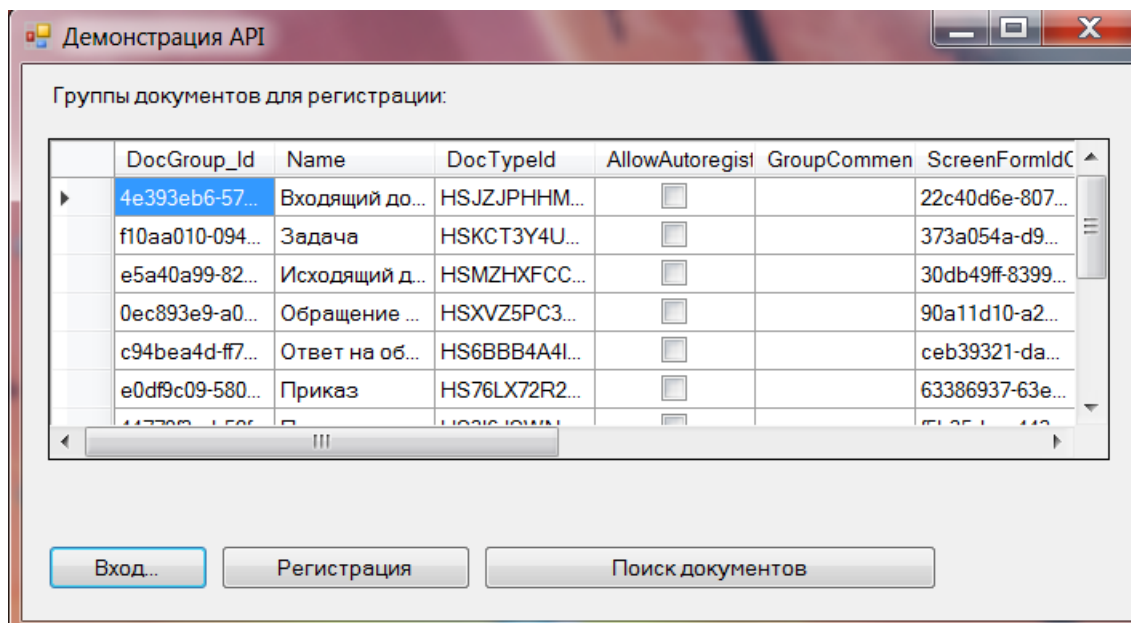


Рис. 4. Окно демонстрационного примера.

Кроме того, в составе демо-примера находятся упомянутые выше сборки Eos.DP.RemoteApi.dll, Eos.DP.Core.dll, Eos.DP Wrapper.dll и wrappers.dll (то же, что Types.dll, для стандартной конфигурации системы).

Работа с пользователями и подключениями

Работа с системой через API возможна только при наличии пользовательского подключения, поэтому для начала работы необходимо вызвать метод интерфейса Connect или ConnectAsWindowsUser. Эти методы возвращают идентификатор созданного сеанса, sessionToken, который необходим для вызова всех остальных методов API. Каждый пользователь может открыть только одно соединение в один момент времени.

С сеансами пользователя связаны дополнительные сущности: блокировки объектов и временные данные.

Блокировки предназначены для эффективной работы в многопользовательской среде. При необходимости изменить существующий в системе объект (документ или запись справочника) на него накладывается блокировка, которая препятствует редактированию объекта кем-либо ещё. При сохранении измененного объекта или отказе от редактирования блокировка снимается.

Временные данные позволяют ассоциировать с текущим пользовательским сеансом блоки двоичных данных. Примером наиболее частого применения таких блоков является добавление файлов в карточку документа. Сначала закачайте двоичные блоки с содержимым файлов, а затем создайте объект – РК документа и добавьте в него ссылки на закачанные файлы. При сохранении такой карточки файлы будут помещены в постоянное хранилище.

Для завершения сеанса работы с системой необходимо вызвать метод `Disconnect`. При этом все блокировки пользователя и несохранённые двоичные блоки будут удалены.

Connect

```
byte[] Connect(string user, string password, bool forceReconnect)
```

Осуществляет подключение к системе как пользователь с логином и паролем.

Параметры:

<code>user:</code>	логин пользователя, под которым производится подключение
<code>password:</code>	пароль этого пользователя
<code>forceReconnect:</code>	поведение при существующем подключении

Примечание:

Если в системе уже существует сессия с этим пользователем, то поведение задаётся параметром `forceReconnect`. Если `forceReconnect = true`, предыдущая сессия будет принудительно завершена, в противном случае вызов `Connect` завершится с ошибкой.

Возвращаемое значение:

Идентификатор соединения, который необходимо сохранить. При использовании других методов API требуется указывать это значение в качестве параметра `sessionToken`.

ConnectAsWindowsUser

```
byte[] ConnectAsWindowsUser(bool forceReconnect)
```

Осуществляет подключение к системе как текущий пользователь Windows.

Параметры:

<code>forceReconnect:</code>	поведение при существующем подключении
------------------------------	--

Если в системе уже существует сессия с этим пользователем, то поведение задаётся параметром `forceReconnect`. Если `forceReconnect = true`, предыдущая сессия будет принудительно завершена, в противном случае вызов `ConnectAsWindowsUser` завершится с ошибкой.

Возвращаемое значение:

Идентификатор соединения, который необходимо сохранить. При использовании других методов API требуется указывать это значение в качестве параметра `sessionToken`.

Disconnect

```
void Disconnect(byte[] sessionToken)
```

Завершает сеанс пользователя в системе.

Параметры:

<code>sessionToken:</code>	идентификатор соединения
----------------------------	--------------------------

GetCurrentUser

```
DataSet GetCurrentUser(byte[] sessionToken)
```

Возвращает описание пользователя, под которым выполнено подключение.

Параметры:

<code>sessionToken:</code>	идентификатор соединения
----------------------------	--------------------------

Возвращаемое значение:

Буфер объекта «Пользователь» (`Eos.DP.Wrapper.User`) в формате `DataSet`.

SaveRecentDocument

```
void SaveRecentDocument(byte[] sessionToken, Guid documentId,
    string documentName)
```

Добавляет документ в список последних используемых документов.

Параметры:

<code>sessionToken:</code>	идентификатор соединения
<code>documentId:</code>	идентификатор используемого документа
<code>documentName:</code>	Этот параметр не используется

Unlock

Снимает блокировку с объекта.

```
void Unlock(byte[] sessionToken, Guid lockId)
```

Параметры:

<code>sessionToken:</code>	идентификатор соединения
<code>lockId:</code>	идентификатор объекта

Примечание:

При взятии объекта (документа или справочника) на редактирование устанавливается блокировка, предотвращающая другие попытки редактирования этого объекта. При сохранении объекта блокировка снимается автоматически. Вызывайте `Unlock` только в случае отказа от редактирования.

SaveSessionData

Сохраняет бинарный блок данных во временном разделе.

```
void SaveSessionData(SaveSessionMessage message)
```

Параметры:

<code>message</code> – структура со следующими полями:	
<code>byte[] SessionToken:</code>	идентификатор соединения
<code>Guid Id:</code>	идентификатор блока данных
<code>Stream Data:</code>	объект <code>Stream</code> , позволяющий загрузить эти данные

Примечание:

Используйте этот метод для добавления файлов к РК документа. Загрузите один или несколько файлов вызовами `SaveSessionData`, затем добавьте в секцию `Files` документа строки, ссылающиеся на нужные бинарные блоки (поле `File_Id` должно содержать значение, равное полю `message.Id` добавленного файла) и сохраните документ.

Сохранённые с документом блоки автоматически перемещаются из временного раздела в постоянное хранилище. Оставшиеся во временном разделе блоки удаляются при завершении сеанса пользователя.

Пример использования:

```
using (var file = new FileStream(openFileDialog.FileName, FileMode.Open))
{
    var fileRow = DocVersion.AddFilesRow();
```

```

fileRow.File_Id = Guid.NewGuid();
fileRow.FileName = Path.GetFileName(openFileDialog.FileName);
var msg = new SaveSessionMessage();
msg.SessionToken = Token;
msg.Id = fileRow.File_Id;
msg.Data = file;
Api.SaveSessionData(msg);
}

```

LoadSessionData

```
Stream LoadSessionData(byte[] sessionToken, Guid id)
```

Возвращает блок данных из временного раздела.

Параметры:

sessionToken:	идентификатор соединения
id:	идентификатор блока данных

Возвращаемое значение:

Объект Stream для чтения ранее сохранённого блока временных данных.

DeleteSessionData

```
void DeleteSessionData(byte[] sessionToken, Guid id)
```

Удаляет блок данных из временного раздела.

Параметры:

sessionToken:	идентификатор соединения
id:	идентификатор блока данных

GetAuthenticationMode

```
AuthenticationMode GetAuthenticationMode()
```

Возвращает тип аутентификации (Windows или None).

IsDisconnected

```
bool IsDisconnected(byte[] sessionToken)
```

Проверяет валидность сессии.

Параметры:

byte[] sessionToken:	идентификатор соединения
----------------------	--------------------------

Возвращаемое значение:

Возвращает значение True, если сессия с идентификатором sessionToken больше не действительна.

Пример:

```
if (_api.IsDisconnected(_token)) MessageBox.Show("Сессия завершена");
```

GetPrivateFoldersTree

```
DataSet GetPrivateFoldersTree(byte[] sessionToken)
```


Возвращает полное дерево личных папок, которые может увидеть пользователь.

Параметры:

sessionToken: идентификатор соединения

Возвращаемое значение:

Объект DataSet, содержащий единственную таблицу со следующими полями:

Имя поля	Тип	Назначение
<code>_Id</code>	<code>Guid</code>	Идентификатор личной папки
<code>ParentRubric Id</code>	<code>Guid</code>	Идентификатор родительской папки в дереве
<code>OrderNum</code>	<code>int</code>	Значение для сортировки папок в одном узле дерева
<code>DisplayName</code>	<code>string</code>	Название папки
<code>OwnerId</code>	<code>Guid</code>	Идентификатор владельца папки
<code>ShowUnreaded</code>	<code>bool</code>	Если равен True, то для папки выставлен флаг «Показывать количество неп прочтенных документов»
<code>ShowBolded</code>	<code>bool</code>	Если равен True, то для папки выставлен флаг «Показывать папку шрифтом Bold»
<code>IsRoot</code>	<code>bool</code>	Если равен True, то в дереве папка является корневой

Пример. Формирование критерия для поиска документов (все личные папки):

```
if (checkFolder.Checked == true)
{
    // Ищем все документы в личных папках текущего пользователя

    // Получаем в FolderTree дерево личных папок пользователя
    DataSet FolderTree = _api.GetPrivateFoldersTree(_token);

    // В array помещаем список идентификаторов личных папок пользователя
    (поле _Id из полученного DataSet'a)
    List<Guid> array = new List<Guid>();
    foreach (DataRow row in FolderTree.Tables[0].Rows)
    array.Add((Guid)row["_Id"]);

    // Добавляем критерий поиска
    critery.AddParamCritery("doc.Folders.Folder_Id", CriteryType.In,
    DbType.Guid, array.ToArray());
}
```

GetMyACLForFolder

DataSet GetMyACLForFolder(byte[] sessionToken, Guid folderId)

Возвращает права пользователя для личной папки.

Параметры:

sessionToken: идентификатор соединения

folderId: идентификатор личной папки

Возвращаемое значение:

Объект `DataSet`, содержащий таблицу с колонками `_Id` (идентификатор субъекта безопасности), колонками в которых указаны права доступа к документу (`CanRead`, `CanEdit` и т.д.).

```
Guid[] GetUserRoles(byte[] token, Guid userId);
ConnectTokenInfo GetConnectToken(string messagingToken);
ConnectMessageInfo GetMessagingInfo();
DataSet LoadUserSettingsForCustomUser(byte[] token, Guid userId);
void SaveUserSettings(byte[] token, DataSet settings);
DataSet LoadUserSettings(byte[] token);
DataSet GetSessionsList(byte[] sessionToken);
```

Streaming:

```
Stream LoadDocFileWinRT(byte[] sessionToken, Guid documentId, Guid fileId)
```

Работа со справочниками

Справочники системы представлены объектами в сгенерированном Конфигуратором коде, облегчающем работу со структурами данных `DataSet`. Создаваемые пользователем справочники являются потомками объекта "Справочник с рубрикацией" и наследуют не только служебные поля, но и секцию `Rubrics`, хранящую ссылки на универсальный рубрикатор системы. Более подробная информация приведена в Приложении 1.

LoadCatalog

```
DataSet LoadCatalog(byte[] sessionToken, string typeId, Guid id)
```

Загружает для чтения экземпляр справочника выбранного типа.

Параметры:

<code>sessionToken:</code>	идентификатор соединения
<code>typeId:</code>	строковый идентификатор типа справочника
<code>id:</code>	идентификатор экземпляра

Возвращаемое значение:

Объект `DataSet`, имеющий структуру данных выбранного типа.

Примеры использования:

```
// загружает встроенный объект "Наша организация" и выводит на экран
// его название, так как его можно переименовать в интерфейсе

using Eos.DP.Wrapper.Uol;
Guid orgId = UolBase.OurOrganization;
DataSet buffer = api.LoadCatalog(sessionToken, Organization.TypeId, orgId);
Organization org = new Organization(buffer);
MessageBox.Show(org.Name);
```

Определение папок, в которых находится элемент справочника:

```
private void btnRubrics_Click(object sender, EventArgs e)
{
    // Выводим папки, в которых находится элемент справочника
    Guid orgID = (Guid)dgElements.SelectedRows[0].Cells[0].Value;
    DataSet buffer = _api.LoadCatalog(_token, Organization.TypeId, orgID);
    Organization org = new Organization(buffer);
    // в секции Rubrics хранятся папки
    dgRubrics.DataSource = org.Rubrics;
}
```

Определение папки, в которой находится элемент:

```
private void btnShowName_Click(object sender, EventArgs e)
{
    // Выводим название выбранной папки
    Guid rubric_id = (Guid)dgRubrics.SelectedRows[0].Cells[0].Value;
    DataSet buffer = _api.LoadCatalog(_token, Rubric.TypeId, rubric_id);
    Rubric rub = new Rubric(buffer);
    MessageBox.Show(rub.Name);
}
```

Определение всего дерева папок до корневой, в которых находится элемент справочника:

```
private string tree;
private void MakeTree(Guid rubric_id)
{
    DataSet buffer = _api.LoadCatalog(_token, Rubric.TypeId, rubric_id);
    Rubric rub = new Rubric(buffer);
    tree = rub.Name + Char.ConvertFromUtf32(13) + tree;

    // Вызываем себя, пока не дойдём до корневой папки
    if (rub.ParentRubric_Id != Guid.Empty) MakeTree(rub.ParentRubric_Id);
}

private void btnTree_Click(object sender, EventArgs e)
{
    // Выводим всё дерево папок до корневой для выбранной папки
    Guid rubric_id = (Guid)dgRubrics.SelectedRows[0].Cells[0].Value;
    tree = "";
    // выполняем рекурсивную функцию
    MakeTree(rubric_id);

    MessageBox.Show(tree);
}
```

LoadCatalogForUpdate

```
DataSet LoadCatalogForUpdate(byte[] sessionToken, string typeId, Guid id)
```

Загружает для редактирования экземпляр справочника выбранного типа.

Параметры:

sessionToken:	идентификатор соединения
typeId:	строковый идентификатор типа справочника
id:	идентификатор экземпляра

Возвращаемое значение:

Объект DataSet, имеющий структуру данных выбранного типа.

Примечание:

Метод `LoadCatalogForUpdate` устанавливает блокировку на редактируемом объекте. Для снятия блокировки при отказе от изменений используйте метод `Unlock`.

SaveCatalog

```
void SaveCatalog(byte[] sessionToken, DataSet buffer)
```

Сохраняет, добавляет или удаляет запись справочника

Параметры:

<code>sessionToken:</code>	идентификатор соединения
<code>buffer:</code>	<code>DataSet</code> , содержащий изменения

Примечание:

Объект `buffer` должен иметь правильную структуру сохраняемого справочника. Кроме того, у него должны быть правильно заполнены служебные поля (идентификатор типа и т.д.). Для автоматического выполнения этих требований рекомендуется использовать сгенерированные Конфигуратором объекты из сборки `Types.dll`.

Пример использования:

```
//Создаём новую рубрику документов
var rubric = new Eos.DP.Wrapper.Rubric();
rubric.CreateBuffer(true);
rubric.Name = "Новая рубрика!";

// устанавливаем родительской корневую рубрику "Документы"
rubric.ParentRubric_Id = Eos.DP.Wrapper.Rubric.Documents;

// сохраняем новую рубрику
api.SaveCatalog(sessionToken, rubric.Buffer);
```

SearchCatalog

Выполняет поиск записей справочника.

```
DataSet SearchCatalog(byte[] sessionToken, string typeId,
    string searchExpression, params object[] values)
```

Параметры:

<code>sessionToken:</code>	идентификатор соединения
<code>typeId:</code>	тип справочника (строковый идентификатор)
<code>searchExpression:</code>	строка поискового запроса (см. Приложение 2)
<code>values:</code>	список параметров переменной длины

Возвращаемое значение:

Объект `DataSet`, содержащий таблицу результатов поиска в таблице “Root”. Столбцы этой таблицы определяются из поискового запроса, а каждая строка таблицы представляет собой запись справочника, удовлетворяющую поисковым критериям. Также объект

содержит таблицу “SearchResultOptions”, в которой указано количество записей, удовлетворяющих поисковому запросу.

Примечание:

Формат строки поискового запроса описан в Приложении 2.

Пример использования:

```
// Найти все записи справочника "Граждане" (Контакт),  
// относящиеся к городу Уфе, и отобразить их в таблице dataGrid1  
  
string typeId = Eos.DP.Wrapper.Uol.Citizen.TypeId;  
string searchExpr = "_Id,DisplayName,Town:=(Town,?)";  
var dataSet = api.SearchCatalog(token, typeId, searchExpr, "Уфа");  
dataGrid1.DataSource = dataSet;
```

ExportTypeSchema

Возвращает поток, содержащий схему данных (XSD) для соответствующего типа документа или справочника.

```
Stream ExportTypeSchema(byte[] sessionToken, string typeId)
```

Параметры:

sessionToken:	идентификатор соединения
typeId:	тип справочника (строковый идентификатор)

Возвращаемое значение:

Объект Stream, содержащий схему формате XSD, описывающую структуру данных соответствующего типа документа или справочника.

Пример применения:

Показать все записи выбранного справочника.

```
private void btnLoadCatalog_Click(object sender, EventArgs e)  
{  
    string typeId = (string)catalogGrid.SelectedRows[0].Cells[0].Value;  
  
    // Загружаем схему, описывающую структуру данных справочника  
    Stream schema = Api.ExportTypeSchema(Token, typeId);  
  
    DataSet ds = new DataSet();  
    ds.ReadXml(schema);  
  
    // формируем поисковый запрос, содержащий все поля справочника  
    string searchstring = "";  
    for (int i = 0; i < ds.Tables["Root"].Columns.Count; i++)  
        searchstring = searchstring + ds.Tables["Root"].Columns[i].ColumnName +  
        ",";  
  
    searchstring = searchstring.Substring(0, searchstring.Length - 1);  
    searchstring = searchstring + ":";
```

```
// загружаем справочник и отображаем
LoadCatalog catalogForm = new LoadCatalog(Api, Token, typeId, searchstring);
catalogForm.ShowDialog();
}
```

GetCatalogList

Возвращает список доступных справочников.

```
DataSet SearchCatalog(byte[] sessionToken)
```

Параметры:

```
sessionToken:      идентификатор соединения
```

Возвращаемое значение:

Объект DataSet, содержащий таблицу с названиями и именами типов доступных справочников.

Примечание:

Текущий пользователь должен обладать ролью Технолога.

LoadCatalogRule

Загружает настройки для данного справочника.

```
string LoadCatalogRule(byte[] sessionToken, string typeId, bool forUpdate)
```

Параметры:

```
sessionToken      : идентификатор соединения
typeId            : тип справочника
forUpdate        : для редактирования
```

Возвращаемое значение:

Строка, содержащая xml код с настройками для заданного справочника.

Примеры использования:

Отображаемое имя для справочника.

```
private void btnDictSettings_Click(object sender, EventArgs e)
{
    Guid orgID = (Guid)dgElements.SelectedRows[0].Cells[0].Value;
    // Загружаем настройки для данного справочника (Организация)
    string CatalogRule = _api.LoadCatalogRule(_token, Organization.TypeId,
false);

    XDocument xDoc = XDocument.Parse(CatalogRule);
    var catalog_rule = xDoc.Descendants("catalog_rule").Where(d =>
d.Attribute("RootRubricId") != null).FirstOrDefault();
    var defaultAttr = xDoc.Root.Attribute("DefaultSearchAttributeId").Value;
    var SearchAttributes = catalog_rule.Descendants("search_attribute").Where(d
=> d.Attribute("Id").Value == defaultAttr).ToList();
}
```

```

    MessageBox.Show("Отображаемое имя: " +
SearchAttributes[0].Attribute("Name").Value.ToString());
}

```

Значение этого поля можно использовать в командах SearchCatalog, а также для поиска отображаемого имени в буфере справочника.

Реквизиты уникальности для справочника.

Справочники могут иметь настройки уникальности, при этом эти настройки задаются динамически технологом. Реквизиты уникальности хранятся в настройках справочника как массив UniqueAttributes массивов AttributesCombination с полем ElementNameEng.

К примеру, если элементы справочника должны быть уникальны по атрибуту «Name» – в настройках справочника будет 1 запись в массиве UniqueAttributes, в ней будет 1 запись в массиве AttributesCombination с полем ElementNameEng = «Name». Если элементы справочника должны быть уникальны по комбинации атрибутов «Name» и «Color», а также по атрибуту «Type» – в настройках справочника будет 2 записи в массиве UniqueAttributes, в первой будет 2 записи в AttributesCombination («Name», «Color») – а во второй – 1 запись («Type»).

```

private void btnUniqueAttr_Click(object sender, EventArgs e)
{
    // Получаем реквизиты уникальности для выбранного справочника

    // Получим тип выбранного справочника
    string TypeID = (string)catalogGrid.SelectedRows[0].Cells[0].Value;

    // Загружаем настройки выбранного справочника
    var CatalogRule = _api.LoadCatalogRule(_token, TypeID, false);
    XDocument xDoc = XDocument.Parse(CatalogRule);
    var unique_elements = xDoc.Descendants("unique_element").ToList();

    // Выводим все комбинации атрибутов уникальности
    foreach (var element in unique_elements)
    {
        MessageBox.Show(element.Element("DisplayName").Value);
    }
    if (unique_elements.Count == 0) MessageBox.Show("Атрибуты уникальности не
указаны");
}

```

Unlock

Снимает блокировку с объекта.

```
void Unlock(byte[] sessionToken, Guid lockId)
```

Параметры:

sessionToken:	идентификатор соединения
lockId:	идентификатор объекта

Примечание:

Функция используется только для работы со справочниками, для работы с документами пользуйтесь функциями UnlockRecordsForUpdate и UnlockRecordForInsertChild. При взятии справочника на редактирование устанавливается блокировка, предотвращающая

другие попытки редактирования этого объекта. При сохранении объекта блокировка снимается автоматически. Вызывайте `Unlock` только в случае отказа от редактирования.

GetMyACLForCatalog

```
DataSet GetMyACLForCatalog(byte[] sessionToken, string typeId)
```

Возвращает права текущего пользователя для справочника.

Параметры:

```
sessionToken: идентификатор соединения  
typeId: строковый идентификатор типа справочника
```

Возвращаемое значение:

Объект `DataSet`, содержащий несколько таблиц, из которых играющую роль и имеющую необходимые данные является *CatalogRights*. В ее колонках, имеющих префикс *Can* (*CanRead*, *CanEdit* и т.д.), указаны права доступа к справочнику. Данные в остальных колонках не имеют значения.

Пример использования:

Выводим все права текущего пользователя к выбранному справочнику.

```
// Получаем права текущего пользователя на справочник  
  
// Получим тип выбранного справочника  
stringTypeID = (string)catalogGrid.SelectedRows[0].Cells[0].Value;  
  
DataSet MyACL = _api.GetMyACLForCatalog(_token, typeid);  
string mes = "";  
  
for (int c = 10; c <= 18; c++)  
{  
    mes += MyACL.Tables["CatalogRights"].Columns[c].ColumnName.ToString() + " = "  
    +  
    MyACL.Tables["CatalogRights"].Rows[0].ItemArray[c].ToString() +  
    Char.ConvertFromUtf32(13);  
}  
MessageBox.Show(mes);
```

ReplaceCatalogReferences

```
DataSet ReplaceCatalogReferences(byte[] token, string catalogTypeId, Guid[] oldIds,  
Guid newId);
```

Заменяет записи, имеющие идентификаторы *oldIds*, справочника типа *catalogTypeId*, на запись *newId*, встречающиеся в сущностях (справочниках, документах) в качестве ссылок. Записи *oldIds* удаляет из справочника.

Возвращает несколько таблиц, содержащих отчет о проделанной операции.

CreateCatalog

```
DataSet CreateCatalog(byte[] sessionToken, string typeId);
```


Создает запись типа *typeId* справочника. Возвращает данные этой записи.

GetPrivateFoldersTree

```
DataSet GetPrivateFoldersTree(byte[] sessionToken);
```

Возвращает папки текущего пользователя. Возвращаемый DataSet состоит из нескольких таблиц, основная из которых Root.

Таблица Root содержит столбцы:

_Id: идентификатор папки;

ParentRubric_Id: идентификатор родительской папки;

OrderNum: порядок сортировки;

Name: название папки;

IsShared: принадлежит ли папка текущему пользователю;

OwnerId: идентификатор пользователя папки;

UnreadMode: константа-настройка, количество чего показывать рядом с папкой;

ShowBoded: выделять ли папку жирным стилем;

CanRead: разрешен ли доступ к папке;

IsUserDeleted: является ли владелец папки скрытым (удаленным).

GetMyACLForFolder

```
DataSet GetMyACLForFolder(byte[] sessionToken, Guid folderId);
```

Возвращает права текущего пользователя для папки с идентификатором *folderId*.

Возвращаемое значение:

Объект DataSet, содержащий несколько таблиц, из которых играющую роль и имеющую необходимые данные является ACL. В ее колонках, имеющих префикс *Can* (*CanRead*, *CanEdit* и т.д.), указаны права доступа к папке и документам. Данные в остальных колонках не имеют значения.

Работа с документами

Документы также представлены объектами в сгенерированном коде. Главная конструктивная особенность документов – наличие секции *Versions*, которая хранит все версионные реквизиты документа (дату, номер, аннотацию, секции файлов, рубрики, ссылки и любые добавляемые в конфигураторе поля). Модифицировать можно только версию, в которой установлен признак *IsActual*.

Вот псевдокод для нахождения такой версии в объекте document:

```
var docVersion = document.Versions.Single(x => x.IsActual == true);  
docVersion.Date = DateTime.Now;
```

Работа с документами средствами API отражается в протоколе.

CreateDocumentByGroup

```
DataSet CreateDocumentByGroup(byte[] sessionToken, Guid groupId)
```

Создаёт объект – документ, принадлежащий группе *groupId*.

Параметры:

sessionToken:	идентификатор соединения
groupId:	идентификатор группы документов

Возвращаемое значение:

Объект DataSet, описывающий структуру документа нужного типа (определённого на группе документов).

Примечание:

Можно создавать только те документы, на регистрацию которых текущему пользователю выданы права. Для регистрации документа в системе полученный объект необходимо сохранить, вызвав метод SaveDocument.

Чтобы получить список групп документов, доступных для регистрации, вызовите метод GetRegistrationDocumentGroups.

Пример использования:

```
// предполагается, что myGroup ссылается на группу входящих документов
DataSet buffer = api.CreateDocumentByGroup(token, myGroup);
var document = new Eos.DP.UserDefinedTypes.IncomingDoc(buffer);
var docVersion = document.Versions.Single(x => x.IsActual == true);
docVersion.Date = DateTime.Now;
docVersion.Number = "Исх. № 1/2008";
api.SaveDocument(document.Buffer);
```

CreateDocumentByGroupAndOther

```
DataSet CreateDocumentByGroupAndOther(byte[] sessionToken, Guid groupId, Guid
rootId, Guid linkTypeId, Guid scriptId);
```

Создаёт объект – документ, принадлежащий группе groupId. Проставляет ему ссылку типа linkTypeId на документ, представленный идентификатором rootId.

Параметры:

sessionToken:	идентификатор соединения
groupId:	идентификатор группы документов
rootId:	документ, на который проставляется ссылка
linkTypeId:	тип ссылки, которая установится между создаваемым документом и документом rootId
scriptId:	id скрипта, получаемый из группы Root документа (на который проставляется ссылка), применяемый на создаваемом документе

Возвращаемое значение:

Объект DataSet, аналогичный возвращаемому значению метода CreateDocumentByGroup.

CreateNewDocumentVersion

```
DataSet CreateNewDocumentVersion(byte[] sessionToken, Guid documentId,
Guid versionId)
```

Создаёт новую версию документа – копию существующей версии.

Параметры:

sessionToken:	идентификатор соединения
documentId:	идентификатор документа
versionId:	идентификатор версии, на основании которой делается копия

Возвращаемое значение:

Объект DataSet, содержащий запрошенный документ с созданной новой версией (копией версии `versionId`).

Примечание:

Параметр `versionId` должен быть равен полю `_Id` той версии документа, копию которой необходимо создать. В полученном объекте активной будет новая версия.

На документ `documentId` создаётся блокировка редактирования. Для снятия блокировки и сохранения новой версии в БД вызовите метод `SaveDocument`. В случае отказа от сохранения документа следует вызвать `Unlock`.

Пример использования:

```
// Создать новую версию существующего документа doc
// на базе версии с номером 3
// и добавить в аннотацию строку "Измененная редакция".
// У пользователя предполагается наличие права "управление версиями".

// Получаем строку версии 3
var docVersion = doc.Versions.Single(x => x.VersionNumber == 3);

// Создаём новую версию документа
var buffer = api.CreateNewDocumentVersion(token, doc._Id, docVersion._Id);
var newDoc = new Eos.DP.Wrapper.Doc.BaseDocument(buffer);

// Находим активную (созданную) версию и модифицируем её
var newVersion = newDoc.Versions.Single(x => x.IsActual == true);
newVersion.ANNOTATION = newVersion.ANNOTATION + ". Измененная редакция."

// Сохраняем документ с новой версией
api.SaveDocument(newDoc.Buffer);
```

ReleaseNewDocument

```
void ReleaseNewDocument(byte[] sessionToken, DataSet wrapper)
```

Освобождает зарезервированные документом номера нумератора группы.

Параметры:

`sessionToken`: идентификатор соединения
`wrapper`: буфер документа, создание которого отменяется

GetRegistrationDocumentGroups

```
DataSet GetRegistrationDocumentGroups(byte[] sessionToken)
```

Возвращает список групп документов, доступных для регистрации текущему пользователю.

Параметры:

`sessionToken`: идентификатор соединения

Возвращаемое значение:

DataSet, содержащий таблицу следующей структуры:

Guid DocGroup_Id:	идентификатор группы документов
-------------------	---------------------------------

string Name:	наименование группы документов
--------------	--------------------------------

Каждая строка этой таблицы представляет собой одну доступную для регистрации группу документов.

LoadDocument

DataSet LoadDocument(byte[] sessionToken, Guid documentId)
--

Загружает для чтения документ с идентификатором documentId.

Параметры:

sessionToken:	идентификатор соединения
documentId:	идентификатор документа

Возвращаемое значение:

Объект DataSet, содержащий структуру запрошенного документа (в зависимости от его типа).

Примечание:

Факт чтения документа фиксируется в протоколе системы.

Примеры использования:

```
// Вывести на экран номер документа с идентификатором docId.
// Предполагается, что он доступен текущему пользователю для чтения

DataSet buffer = api.LoadDocument(token, docId);
var document = new Eos.DP.Wrapper.Doc.BaseDocument(buffer);
var docVersion = document.Versions.Single(x => x.IsActual == true);
MessageBox.Show("Номер документа: " + docVersion.Number);
```

Загрузить и отобразить все поручения документа. В загруженном документе просматриваем секцию Tasks.

```
private void LoadTasks(Guid dId)
{
    BaseDocument doc = new BaseDocument(Api.LoadDocument(Token, DocId));
    DocVersion = doc.Versions.Single(x => x.IsActual == true);

    foreach (var row in DocVersion.Tasks)
    {
        // загрузить и отобразить документ с row.TaskId
        TaskForm tf = new TaskForm(Api, Token, Guid.Empty, row.TaskId);
        tf.ShowDialog();
    }
}
```

```
private void ViewDocument()
{
    Task doc = new Task(Api.LoadDocument(Token, DocId));
    DocVersion = doc.Versions.Single(x => x.IsActual == true);
    textNumber.Text = DocVersion.Number;
    textDate.Text = DocVersion.Date.ToString();
    textStatusName.Text = DocVersion.DocumentStatus_Name;
    textUrgencyName.Text = DocVersion.TaskUrgencyName;
    checkKonf.Checked = DocVersion.IsConfidential;
}
```

Загрузить все отчёты к поручению. Если значение ParentDocId в секции Performers не пустое – значит существует отчёт от исполнителя.

```
private void LoadReports(Guid tId)
{
    Task doc = new Task(Api.LoadDocument(Token, tId));
    DocVersion = doc.Versions.Single(x => x.IsActual == true);

    foreach (var row in DocVersion.Performers)
        if (row.ParentDocId != Guid.Empty)
        {
            ReportForm rf = new ReportForm(Api, Token, row.ParentDocId);
            rf.ShowDialog();
        }
}
```

```
private void ViewReport()
{
    BaseDocument doc = new BaseDocument(Api.LoadDocument(Token, DocId));
    DocVersion = doc.Versions.Single(x => x.IsActual == true);

    textNumber.Text = DocVersion.Number.ToString();
    textAnnotation.Text = DocVersion.ANNOTATION;
}
```

LoadDocumentForUpdate

```
DataSet LoadDocumentForUpdate(byte[] sessionToken, Guid documentId)
```

Загружает для редактирования документ с идентификатором documentId.

Параметры:

sessionToken:	идентификатор соединения
documentId:	идентификатор документа

Возвращаемое значение:

Объект DataSet, содержащий структуру запрошенного документа (в зависимости от его типа).

Примечание:

Вызов LoadDocumentForUpdate приводит к установке блокировки редактирования. Для её снятия необходимо вызвать SaveDocument или Unlock.

Факт чтения документа фиксируется в протоколе.

SaveDocument

```
void SaveDocument(byte[] sessionToken, DataSet buffer)
```

Сохраняет, удаляет или создаёт новый документ системы.

Параметры:

sessionToken:	идентификатор соединения
buffer:	DataSet документа нужной структуры

Примечание: для корректного сохранения нового документа необходимо правильно заполнить служебные поля, таблицу версий и права доступа. Проще всего создать правильно заполненный буфер можно получить, вызвав метод CreateDocumentByGroup. Для удаления документа необходимо его загрузить, вызвать метод Delete и сохранить изменённый объект.

Пример использования:

```
// Удалить документ с идентификатором docId

var buffer = api.LoadDocument(token, docId);
var doc = new Eos.DP.Wrapper.Doc.BaseDocument(buffer);
doc.Delete();

// Сохраняем объект с пометкой удаления
api.SaveDocument(token, doc.Buffer);
```

SearchDocsByCriteries

```
DataSet SearchDocsByCriteries(byte[] sessionToken, string typeId,
    CriteryGroup criteries, Pagination pagination, Targets targets)
```

Выполняет поиск документов по набору критериев.

Параметры:

sessionToken:	идентификатор соединения
typeId:	идентификатор типа документов для поиска
criteries:	список критериев, объединённых условиями AND и OR
pagination:	структура, содержащая следующие поля, либо Pagination.Empty
PageNum:	номер требуемой страницы
PageSize:	количество записей в каждой странице
targets:	структура, содержащая следующие поля, либо Targets.Empty
Fields:	массив строк, определяющий какие атрибуты документа мы хотим получить в результате запроса. Синтаксис представления атрибутов аналогичен таковому в критериях запроса.
Orders:	массив структур, определяющий порядок сортировки (или ее отсутствие) по каждому из запрашиваемых атрибутов документа

Возвращаемое значение:

Объект DataSet, содержащий таблицу "Root" следующего содержания:

```
_Id:                идентификатор документа
<имя атрибута документа, указанного targets.Fields>: значение атрибута документа, указанного в массиве Fields параметра targets
...
```

Возвращаемый DataSet содержит только одну таблицу, которая может содержать только атрибуты головной секции.

Пример использования:

```
private void FindDocs()
{
    CriteryGroup critery = new CriteryGroup();
    if (DocNum.Text.Trim() != string.Empty)
        critery.AddTextCritery("doc.Versions.Number", DocNum.Text, true);
    AddDateRange(critery, "doc.Versions.Date", DateFrom.Value, DateTo.Value);
    if (textFolder.Text.Trim() != string.Empty)
        critery.AddTextCritery("doc.Folders.Folder.Name", textFolder.Text, true);

    if (checkFolder.Checked == true)
    {
        // Ищем все документы в личных папках текущего пользователя
        // Получаем в FolderTree дерево личных папок пользователя
        DataSet FolderTree = _api.GetPrivateFoldersTree(_token);
    }
}
```

```

        // В array помещаем список идентификаторов личных папок пользователя
        (поле _Id из полученного DataSet'a)
        List<Guid> array = new List<Guid>();
        foreach (DataRow row in FolderTree.Tables[0].Rows)
            array.Add((Guid) row["_Id"]);

        // Добавляем критерий поиска
        critery.AddParamCritery("doc.Folders.Folder_Id", CriteryType.In,
        DbType.Guid, array.ToArray());
    }
    else
        // Ищем все документы в заданной личной папке пользователя
        if (textFolder.Text.Trim() != string.Empty)
            critery.AddTextCritery("doc.Folders.Folder.Name", textFolder.Text,
            true);

        if (Content.Text.Trim() != string.Empty)
            critery.AddFullTextCritery("doc.Versions.Files.File.CONTENT",
            Content.Text, true);

        var targets = new Targets(new[] { "doc.CreationDate", "doc.EditDate" },
        new[] { System.Data.SqlClient.SortOrder.Ascending,
        System.Data.SqlClient.SortOrder.Descending });

        // Выводим найденные документы в таблицу
        SearchResultGrid.DataSource = _api.SearchDocsByCriteries(_token,
        BaseDocument.TypeId, critery, Pagination.Empty, targets);
        SearchResultGrid.DataMember = "Root";
    }
}

```

Примечание:

При поиске по версионным реквизитам (как в примере) поиск выполняется только по содержимому активной версии. Результаты поиска ограничены правами текущего пользователя на документы системы.

LoadDocsByCriteries

```

DataSet LoadDocsByCriteries(byte[] sessionToken, string typeId, CriteryGroup
criteries, Pagination pagination, Targets targets)

```

Выполняет поиск и загрузку документов по критериям.

Параметры:

sessionToken:	идентификатор соединения
typeId:	идентификатор типа документов для поиска
criteries:	список критериев, объединённых условиями AND и OR
pagination:	структура, содержащая следующие поля, либо Pagination.Empty
PageNum:	номер требуемой страницы
PageSize:	количество записей в каждой странице
targets:	структура, содержащая следующие поля, либо Targets.Empty
Fields:	массив строк, определяющий какие атрибуты документа мы хотим получить в результате запроса. Синтаксис представления атрибутов аналогичен таковому в критериях запроса.
Orders:	массив структур, определяющий порядок сортировки (или ее отсутствие) по каждому из запрашиваемых атрибутов документа

Возвращаемое значение:

Объект DataSet, содержащий данные всех найденных документов, структурно соответствующий указанному типу документа.

При загрузке документов через LoadDocsByCriteriaes, загрузка документов не протоколируется.

Пример:

```
private void FindDocs()
{
    CriteryGroup critery = new CriteryGroup();

    if (DocNum.Text.Trim() != string.Empty)
        critery.AddTextCritery("doc.Versions.Number", DocNum.Text, true);

    AddDateRange(critery, "doc.Versions.Date", DateFrom.Value,
DateTo.Value);

    if (textFolder.Text.Trim() != string.Empty)
        critery.AddTextCritery("doc.Folders.Folder.Name",
textFolder.Text, true);

    if (checkFolder.Checked == true)
    {
        // Ищем все документы в личных папках текущего пользователя

        // Получаем в FolderTree дерево личных папок пользователя
        DataSet FolderTree = _api.GetPrivateFoldersTree(_token);

        // В array помещаем список идентификаторов личных папок
        пользователя (поле _Id из полученного DataSet'a)
        List<Guid> array = new List<Guid>();
        foreach (DataRow row in FolderTree.Tables[0].Rows)
            array.Add((Guid)row["_Id"]);

        // Добавляем критерий поиска
        critery.AddParamCritery("doc.Folders.Folder_Id",
CriteryType.In, DbType.Guid, array.ToArray());
    }
    else
        // Ищем все документы в заданной личной папке пользователя
        if (textFolder.Text.Trim() != string.Empty)
            critery.AddTextCritery("doc.Folders.Folder.Name",
textFolder.Text, true);

        if (Content.Text.Trim() != string.Empty)

            critery.AddFullTextCritery("doc.Versions.Files.File.CONTENT",
Content.Text, true);

            // Получим номер документа и аннотацию из секции Versions
            var targets = new Targets(new[] { "doc.Versions.Number",
"doc.Versions.ANNOTATION"},
new[] { System.Data.SqlClient.SortOrder.Ascending,
System.Data.SqlClient.SortOrder.Descending });

            // Выводим версии найденных документов в таблицу
            SearchResultGrid.DataSource = _api.LoadDocsByCriteriaes(_token,
BaseDocument.TypeId, critery, Pagination.Empty, Targets.Empty);
}
```



```

        SearchResultGrid.DataMember = "Versions";
    }

```

GetMyACLForDocument

```

DataSet GetMyACLForDocument(byte[] SessionToken, Guid documentId, Guid
documentGroupId)

```

Возвращает таблицу, в которой указаны права на указанный документ текущего пользователя.

Параметры:

SessionToken	: идентификатор соединения
documentId	: идентификатор документа
documentGroupId	: идентификатор группы документов

Возвращаемое значение:

Объект DataSet, содержащий несколько таблиц, из которых играющую роль и имеющую необходимые данные является *ACL*. В ее колонках, имеющих префикс *Can* (*CanRead*, *CanEdit* и т.д.), указаны права доступа к документу. Данные в остальных колонках и таблицах не имеют значения.

Пример использования:

Выводим все права доступа текущего пользователя к выбранному документу.

```

private void GetMyACL(Guid dId)
{
    DataSet ACL = _api.GetMyACLForDocument(_token, dId, Guid.Empty);
    string mes = "";
    for (int c = 9; c <= 19; c++)
        mes += ACL.Tables["ACL"].Columns[c].ColumnName.ToString() + " = " +
        ACL.Tables["ACL"].Rows[0].ItemArray[c].ToString() +
        Char.ConvertFromUtf32(13);
    MessageBox.Show(mes);
}

```

GetDocumentLog

```

DataSet GetDocumentLog(byte[] sessionToken, Guid documentId)

```

Возвращает протокол действий над документом.

Параметры:

sessionToken:	идентификатор соединения
documentId:	идентификатор документа

Возвращаемое значение:

Объект DataSet, содержащий таблицу следующей структуры:

DateTime CreationDate:	дата записи протокола
string Name:	имя пользователя – инициатора операции
string DisplayName:	протоколируемая операция
string Comment:	комментарий
int Order:	порядковый номер записи протокола

LoadDocFile

```

DocumentFileMessage LoadDocFile(DocumentFileRequest request)

```

Загружает присоединённый к документу файл.

Параметры:

DocumentFileRequest: структура со следующими полями:	
byte[] SessionToken:	идентификатор соединения
Guid DocumentId:	идентификатор документа
Guid FileId:	идентификатор записи файла в таблице Files

Возвращаемое значение:

DocumentFileMessage: структура со следующими полями:	
string FileName:	имя файла
Stream FileStream:	объект для чтения бинарных данных

GetPrivateSearches

DataSet GetPrivatSearches(byte[] SessionToken)
--

Возвращает список личных поисковых запросов.

Параметры:

byte[] SessionToken:	идентификатор соединения
----------------------	--------------------------

Возвращаемое значение: DataSet, содержащий таблицу с колонками _Id (идентификатор личного поискового запроса), DisplayName (название личного поискового запроса)

RunPrivateSearch

DataSet RunPrivateSearch(byte[] sessionToken, Guid searchId, Pagination pagination, Targets targets)
--

Выполняет личный поисковый запрос.

Параметры:

sessionToken:	идентификатор соединения
searchId:	идентификатор личного поискового запроса
pagination:	структура, содержащая следующие поля, либо Pagination.Empty
PageNum:	номер требуемой страницы
PageSize:	количество записей в каждой странице
targets:	структура, содержащая следующие поля, либо Targets.Empty
Fields:	массив строк, определяющий какие атрибуты документа мы хотим получить в результате запроса. Синтаксис представления атрибутов аналогичен таковому в критериях запроса.
Orders:	массив структур, определяющий порядок сортировки (или ее отсутствие) по каждому из запрашиваемых атрибутов документа

Возвращаемое значение:

Объект DataSet, содержащий таблицу следующего содержания:

_Id:	идентификатор документа, удовлетворяющий личному поисковому запросу
<имя атрибута документа, указанного targets.Fields>:	значение атрибута документа, указанного в массиве Fields параметра targets
...	

AddDocumentsToFolders

```
DataSet AddDocumentsToFolders(byte[] sessionToken, Guid[] docIds, Guid[] folderIds)
```

Добавляет документы в указанные личные папки.

Параметры:

```
sessionToken: идентификатор соединения
docIds       : массив идентификаторов документов
folderIds    : массив идентификаторов личных папок
```

Возвращаемое значение:

Объект DataSet, содержащий информацию о возникших ошибках при действиях с документами.

Примечание:

Функция может использоваться для групповой обработки документов.

Пример использования:

```
private Guid[] _docIds;

public Guid[] DocIds
{
    get { return _docIds; }
    set { _docIds = value; }
}

private void btnAddDocsToFolders_Click(object sender, EventArgs e)
{
    // заполняем список папок
    List<Guid> arrayFolders = new List<Guid>();
    for (var counter = 0; counter < gridPrivateFolders.SelectedRows.Count;
        counter++)
    {
        arrayFolders.Add((Guid)gridPrivateFolders.SelectedRows[counter].Cells[0].Value);
    }

    Guid[] FolderIds = arrayFolders.ToArray();

    // добавляем документы в папки
    Api.AddDocumentsToFolders(Token, DocIds, FolderIds);
}
```

MoveDocumentsFromFolderToFolders

```
DataSet MoveDocumentsFromFolderToFolders(byte[] sessionToken, Guid[] docIds,
Guid sourceFolderId, Guid[] destFolderIds)
```

Перемещает документы из указанной папки в другие.

Параметры:

```
sessionToken    : идентификатор соединения
docIds          : массив идентификаторов документов
sourceFolderId  : идентификатор папки-источника
destFolderIds   : массив идентификаторов папок назначения
```

Возвращаемое значение:

Объект DataSet, содержащий информацию о возникших ошибках при действиях с документами.

Примечание:

Функция может использоваться для групповой обработки документов.

RemoveDocumentsFromFolders

```
DataSet RemoveDocumentsFromFolders(byte[] sessionToken, Guid[] docIds, Guid[] folderIds)
```

Удаляет документы из указанных папок.

Параметры:

sessionToken	: идентификатор соединения
docIds	: массив идентификаторов документов
folderIds	: идентификатор папки-источника

Возвращаемое значение:

Объект DataSet, содержащий информацию о возникших ошибках при действиях с документами.

Примечание:

Функция может использоваться для групповой обработки документов.

LockRecordsForUpdate

Устанавливает блокировку на указанные записи объекта для редактирования.

```
DataSet LockRecordsForUpdate(byte[] sessionToken, DataSet wrapper, Guid[] recordsId)
```

Параметры:

sessionToken:	идентификатор соединения
wrapper	: объект DataSet с данными документа
recordsId	: массив идентификаторов записей

Примечание:

При взятии объекта на редактирование устанавливается блокировка на указанные записи, предотвращающая другие попытки их редактирования. При сохранении объекта блокировка снимается автоматически. Вызывайте [UnlockRecordsForUpdate](#) только в случае отказа от редактирования.

Пример использования:

Изменение записи в секции Authors документа.

```
Doc = Api.LoadDocument(Token, DocId);  
...  
private void btnEdit_Click(object sender, EventArgs e)
```

```

{
// получаем массив идентификаторов записей, которые хотим менять
List<Guid> array = new List<Guid>();
array.Add((Guid)gridAuthors.CurrentRow.Cells[0].Value);
Guid[] arr = array.ToArray();

// блокируем записи для изменения
Api.LockRecordsForUpdate(Token, Doc, array.ToArray());

try
{
BaseDocument doc = new BaseDocument(Doc);
DocVersion = doc.Versions.Single(x => x.IsActual == true);

// Наши изменения
foreach (var row in DocVersion.Authors)
if (row._Id == (Guid)gridAuthors.CurrentRow.Cells[0].Value)
{
    row.Author_Id = (Guid)gridAllAuthors.CurrentRow.Cells[0].Value;
    row.AuthorDisplayName = gridAllAuthors.CurrentRow.Cells[1].ToString();
}
// Сохраняем документ, снимаем блокировки
Api.SaveDocument(Token, Doc);
}
catch
{
// по каким-то причинам редактирование не удалось - нужно снять блокировки
Api.UnlockRecordsForUpdate(Token, Doc, array.ToArray());
}
}

```

UnlockRecordsForUpdate

Устанавливает блокировку на указанные записи объекта для редактирования.

```
DataSet LockRecordsForUpdate(byte[] sessionToken, DataSet wrapper, Guid[] recordsId)
```

Параметры:

sessionToken:	идентификатор соединения
wrapper	: объект DataSet с данными документа
recordsId	: массив идентификаторов записей

Примечание:

При взятии объекта на редактирование устанавливается блокировка на указанные записи, предотвращающая другие попытки их редактирования. При сохранении объекта блокировка снимается автоматически. Вызывайте UnlockRecordsForUpdate только в случае отказа от редактирования.

LockRecordsForInsertChild

Устанавливает блокировку на указанные записи объекта для добавления дочерних элементов.

```
DataSet LockRecordsForUpdate(byte[] sessionToken, DataSet wrapper, Guid recordId)
```

Параметры:

sessionToken:	идентификатор соединения
wrapper	: объект DataSet с данными документа
recordId	: идентификатор записи родительской секции для добавляемых записей

Примечание:

При взятии объекта на редактирование устанавливается блокировка на указанную запись, предотвращающая другие попытки изменения дочерних секций. При сохранении объекта блокировка снимается автоматически. Вызывайте UnlockRecordsForInsertChild только в случае отказа от редактирования.

Пример использования:

Добавляем запись об авторе.

```
Doc = Api.LoadDocument(Token, DocId);

...

private void btnAdd_Click(object sender, EventArgs e)
{
    /* Будем добавлять запись в секцию Authors,
    * для этого ставим блокировку на родительскую запись для секции.
    * В нашем случае - это текущая версия*/

    BaseDocument doc = new BaseDocument(Doc);
    var docVersion = doc.Versions.Single(x => x.IsActual == true);

    Api.LockRecordForInsertChild(Token, Doc, docVersion._Id);

    try
    {
        // Наши изменения
        BaseDocument_Version_Author row = docVersion.AddAuthorsRow();
        row.Author_Id = (Guid)gridAllAuthors.CurrentRow.Cells[0].Value;
        row.AuthorDisplayName = gridAllAuthors.CurrentRow.Cells[1].ToString();

        // Сохраняем документ, снимаем блокировку
        Api.SaveDocument(Token, Doc);
    }
    catch
    {
        // по каким-то причинам редактирование не удалось - нужно снять блокировки
        Api.UnlockRecordForInsertChild(Token, Doc, DocVersion._Id);
    }
}
```

UnlockRecordsForInsertChild

Снимает блокировку с указанной записи объекта, установленную для добавления дочерних элементов.

DataSet LockRecordsForUpdate(byte[] sessionToken, DataSet wrapper, Guid recordId)

Параметры:

sessionToken:	идентификатор соединения
wrapper :	объект DataSet с данными документа
recordId :	идентификатор записи родительской секции для добавляемых записей

Примечание:

При взятии объекта на редактирование устанавливается блокировка на указанную запись, предотвращающая другие попытки изменения дочерних секций. При сохранении объекта блокировка снимается автоматически. Вызывайте `UnlockRecordsForInsertChild` только в случае отказа от редактирования.

GetDocumentPrintData

<code>DocumentPrintMessage</code> GetDocumentPrintData(<code>DocumentPrintRequest</code> req)
--

Возвращает файл в виде потока данных для печати документа в формате PDF.

Параметры:

Req: структура данных с запросом данных для печати
--

`DocumentPrintRequest` - структура со следующими полями:

<code>byte[]</code> SessionToken:	идентификатор соединения
<code>Guid</code> DocumentId:	идентификатор документа

Возвращаемое значение:

Объект `DocumentPrintMessage` со следующей структурой:

<code>string</code> PrintName:	имя файла для печати
<code>string</code> PrintExtension:	расширение файла для печати
<code>Stream</code> PrintStream:	поток, содержащий файл для печати

SaveDocFile

<code>void</code> SaveDocFile(<code>SaveSessionMessage</code> message)

Сохраняет файл.

Параметры:

message – объект класса со следующими свойствами:

<code>Guid</code> sessionToken:	идентификатор соединения
<code>Guid</code> Id: id,	по которому необходимо сохранить файл (на вызывающей метод стороне обычно генерируется перед вызовом метода, сохраняется для записи в документ)
<code>Stream</code> Data:	файловый поток

Работа с оповещениями

CreateEventSubscription

```
void CreateEventSubscription(byte[] sessionToken, string subscriberUri, Guid correlationId, SubscriptionDesc subscriptionDesc);
```

Создает подписки на события системы.

Параметры:

```
byte[] sessionToken: идентификатор соединения
string subscriberUri: адрес оповещения подписчика

Guid correlationId: идентификатор, который
будет передан в оповещении при возникновении события,
предназначен для обеспечения корреляции.
Если correlationId не пустой уведомление будет выполнено
командой Notify, иначе Activate

SubscriptionDesc subscriptionDesc: набор данных определяющих, правила
активации подписки
```

RemoveEventSubscription

```
void RemoveEventSubscription(byte[] sessionToken, string subscriberUri, Guid correlationId, Guid subscriptionDescId);
```

Удаляет подписки на события системы.

Параметры:

```
byte[] sessionToken: идентификатор соединения
string subscriberUri, Guid correlationId: см. описание к методу
CreateEventSubscription.

Guid subscriptionDescId: если передано значение, отличное от Guid.Empty,
то будет удалена только запись с таким значением _Id,
иначе будут удалены все записи с subscriberUri + correlationId.
При отсутствии подписки исключение не возникает.
```

ConfirmEventNotification

```
void ConfirmEventNotification(byte[] sessionToken, params Guid[] idList);
```

Подтверждает получение уведомления о событии системы.

Параметры:

```
byte[] sessionToken: идентификатор соединения
Guid[] idList: список идентификаторов событий, либо исходный список полученных
в уведомлении структур EventInfo
```

SendEventNotification

```
void SendEventNotification(byte[] sessionToken, string subscriberUri, Guid correlationId, string cookie, Guid objectId, string objectType, string eventData);
```

Формирует уведомление об отправке подписчику.

Параметры:

```
byte[] sessionToken: идентификатор соединения
```



```
string subscriberUri: адрес оповещения подписчика
```

Работа с подписью

GetDocumentPassport

```
string GetDocumentPassport(byte[] sessionToken, Guid documentId)
```

Возвращает сериализованный паспорт документа по его идентификатору *documentId*.

Паспорт содержит следующие свойства:

HeaderZone Header: Содержит служебную информацию, необходимую для правильной передачи и интерпретации всего сообщения в целом. Обязательная зона для всех видов сообщений.

DocumentZone Document: Содержит информацию о реквизитах посылаемого документа. Обязательная зона для сообщений о пересылке документа.

TaskZone Task: Содержит информацию о выданных заданиях на исполнение и обработку документа (в виде резолюций и поручений).

ExpansionZone Expansion: Содержит дополнительные, не вошедшие в основную зону, данные из системы отправителя.

AcknowledgementZone Acknowledgement: Содержит ответную информацию о доставке сообщения, об ошибках приема и интерпретации сообщения, о регистрации полученного документа и др. Обязательная зона для сообщений вида уведомление.

CreateDocumentByGroupWithPassport

```
DataSet CreateDocumentByGroupWithPassport(byte[] sessionToken, Guid documentGroupId, byte[] passport)
```

Создает документ по идентификатору группы документов *documentId* и записывает в него данные из сериализованного паспорта *passport*. Метод используется для мобильных решений.

SEVCreatePassportFromDocId

```
string SEVCreatePassportFromDocId(byte[] sessionToken, Guid documentId, Guid uolId)
```

Создает паспорт документа по его идентификатору *documentId*. В информации о выданных заданиях (свойство *TaskList* паспорта) автором (контактом автора) будет являться идентификатор пользователя *uolId*.

SEVCreateDocFromPassport

```
string SEVCreateDocFromPassport(byte[] sessionToken, string passport, Guid groupId)
```

Создает документ по группе документов и паспорту документа, сохраняет его. Возвращает строковое представление идентификатора документа. Метод используется для мобильных решений.

SEVCreateDocFromPassportUnsaved

```
DataSet SEVCreateDocFromPassportUnsaved(byte[] sessionToken, string passport, Guid groupId)
```

Создает документ по группе документов и паспорту документа. Возвращает буфер документа в виде *DataSet*. Метод используется для мобильных решений.

Работа с импортом и экспортом

ExportObject

```
ExportResultMessage ExportObject(UniversalExportRequest request)
```

Экспортирует объект системы: справочник или документ.

Единственным параметром метода является объект класса *UniversalExportRequest*, имеющий следующие свойства:

```
byte[] SessionToken: идентификатор соединения  
string TypeId: тип экспортируемого объекта (справочника или документа)  
Guid Id: идентификатор экспортируемого объекта  
string Settings: сериализованные в xml данные типа TypeSettingsBlock.  
HashSet<Guid> LoadedList: не используется
```

Возвращает объект типа *ExportResultMessage*, содержащий свойства:

```
BuildUniversalTreeLog Log: лог событий  
Stream Tree: экспортируемый поток
```

ImportObject

```
ImportResultMessage ImportObject(UniversalTreeRequest message)
```

Импортирует объект системы: справочник или документ.

Параметром метода является объект класса *UniversalTreeRequest*, имеющий следующие свойства:

```
byte[] SessionToken: идентификатор соединения  
Stream Tree: импортируемый поток данных.  
DataSet ReplacementInfo: объект, содержащий таблицу, в которой прописано, какие id нужно заменить на какие.  
string Settings: сериализованные в xml данные типа ExtendedTypeSettingsBlock.
```

Возвращает объект типа *ImportResultMessage*, содержащий свойство:

```
DataSet ReplacementInfo: то, что заменялось (данные из аргументов)
```

Работа с метаданными

LoadActualMetadata

```
Stream LoadActualMetadata(byte[] sessionToken)
```

Загружает актуальные метаданные из БД.

Работа с пользовательским хранилищем

SaveToCustomStorage

```
void SaveToCustomStorage(CustomStorageMessage message)
```

Сохраняет объект в виде потока данных в БД по указанному ключу.

Параметром выступает объект класса `CustomStorageMessage`, имеющий следующие свойства:

```
byte[] SessionToken: идентификатор соединения  
string Key: ключ сохраняемого объекта  
Stream Data: сохраняемые данные
```

LoadFromCustomStorage

```
Stream LoadFromCustomStorage(byte[] sessionToken, string key)
```

Возвращает ранее сохраненный объект в виде потока данных из БД по указанному ключу *key*.

Работа с настройками системы

LoadSystemSettings

```
DataSet LoadSystemSettings(byte[] sessionToken)
```

Возвращает настройки системы, - *DataSet*, который можно обернуть объектом класса `SystemSettings`.

SaveSystemSettings

```
void SaveSystemSettings(byte[] sessionToken, DataSet settings)
```

Сохраняет настройки системы *settings*

Приложение 1. Стандартные типы системы

Ниже приведен список типов (документов и справочников), находящихся в стандартной поставке eDocLib 2.6. Не приведены некоторые системные (технические) объекты и их поля, носящие служебный характер. Обращение через API к таким полям и объектам не поддерживается и может привести к непредсказуемым результатам.

Имена полей и секций

Описываемые в конфигураторе поля присутствуют с одним и тем же именем как в виде свойств программируемых объектов, так и в виде столбцов DataSet-ов и таблиц БД, хранящих записи документов и справочников. Вы можете использовать API-имена этих полей в строковых выражениях критериев поиска (Приложения 2 и 3).

Добавление в объект простого реквизита с именем *Property* приводит к добавлению в генерируемый код следующих сущностей:

- Свойства *Property* заданного типа;
- Свойства *IsPropertyNull* типа *bool*;
- Метода *SetPropertyNull*, который приводит к установке или сбросу NULL в значении свойства.

Добавление в объект секции с именем *Section* приводит к добавлению в генерируемый код следующих сущностей:

- Свойства *Section*, возвращающего массив объектов типа "дочерняя секция";
- Метода *AddSectionRow*, позволяющего добавить новую строку в дочернюю секцию.

При использовании полей объектов из API учитывайте наследование объектов (описанные на предке поля будут доступны в любом из его потомков).

Пример: справочник "Организация" порождён от справочника "Контакт", поэтому мы можем оперировать полем *DisplayName* при работе со справочником организаций.

Базовые свойства объектов

У всех документов и справочников, генерируемых в коде из Конфигуратора, существуют программные сущности, описываемые ниже:

```
public DataSet CreateBuffer(bool addNewRow);
public void Delete();
public DataSet Buffer { get; set; }
public string GetTypeId();
public DataRow Row { get; set; }
```

- Метод `CreateBuffer(bool)` предназначен для первоначального создания буфера данных объекта. Вызовите этот метод с параметром `true`, если вы создаёте новый экземпляр записи справочника. При загрузке объектов из БД их буфер данных создаётся и заполняется автоматически.
- Метод `Delete()` помечает уже существующий объект для удаления. При сохранении такого объекта он будет удалён из базы данных.
- Функция `GetTypeId()` возвращает строковый идентификатор типа (аналог статического свойства `TypeId`, но для экземпляров).
- Свойство `Row` возвращает объект типа `DataRow`, ссылающийся на строку данных головной секции объекта.
- Свойство `Buffer` возвращает собственно буфер данных, который необходимо передавать в методы сохранения.
- И, наконец, конструктор объекта с параметром типа `DataSet` позволяет осуществить обратное преобразование: создать объект – типизированную "обёртку" над `DataSet`-ом.

Eos.DP Wrapper.Base (Базовый объект)

Базовый тип для всех объектов системы (его поля есть в любом документе и справочнике).

Поле	Тип	Описание
<code>_Id</code>	<code>Guid</code>	Уникальный идентификатор объекта (первичный ключ в БД)
<code>_SecVer</code>	<code>Int</code>	Служебное поле, заполняется автоматически
<code>EditorId</code>	<code>Guid</code>	Идентификатор пользователя, редактировавшего запись последним (заполняется автоматически)
<code>CreatorId</code>	<code>Guid</code>	Идентификатор пользователя, создавшего запись (заполняется автоматически)
<code>EditDate</code>	<code>DateTime</code>	Дата и время последнего изменения (заполняется автоматически)
<code>CreationDate</code>	<code>DateTime</code>	Дата и время создания записи (заполняется автоматически)

Eos.DP Wrapper.Head (Головная секция)

Предок: `Eos.DP Wrapper.Base`. От этого объекта порождены Базовый документ и Справочник.

Поле	Тип	Описание
------	-----	----------

<code>_TypeId</code>	<code>String</code>	Идентификатор типа объекта
----------------------	---------------------	----------------------------

Поле `_TypeId` хранит реальный тип записи объекта (для структур разных типов, имеющих общую таблицу БД, например, документов). Оно заполняется автоматически при вызове `CreateBuffer(true)`.

Линейные справочники системы

Eos.DP.Wrapper.Catalog (Справочник)

Предок всех справочников системы.

Поле	Тип	Описание
Deleted	Bool	Признак логического удаления записи
OrderNum	Int	Порядковый номер (вес) для показа в списках

Eos.DP.Wrapper.IdentityType (Вид удостоверения личности)

Справочник для выбора вида удостоверений личности из списка (паспорт, военный билет и т.д.)

Поле	Тип	Описание
Code	Int	Код удостоверения личности
Type	String	Название вида удостоверения

Eos.DP.Wrapper.AddressType (Тип адреса)

Справочник для выбора видов адресов контактов (юридический, фактический и т.д.)

Поле	Тип	Описание
Code	Int	Код типа адреса
Type	String	Название типа адреса

Eos.DP.Wrapper.LinkType (Тип ссылки)

Справочник для выбора вида ссылки при связывании документов.

Поле	Тип	Описание
LinkName	string	Имя прямой ссылки
BackLinkName	string	Имя обратной ссылки

Eos.DP.Wrapper.Numerator (Нумератор)

Справочник для автоматического назначения номеров документов (счетчик)

Поле	Тип	Описание
NumeratorName	string	Название нумератора
NumeratorValue	Int	Значение счетчика

Eos.DP.Wrapper.MessageType (Тип сообщения)

Справочник типов сообщений. Несмотря на то, что он по умолчанию линейный, он является потомком справочника с рубрикацией.

Поле	Тип	Описание
Name	string	Наименование

Eos.DP.Wrapper.ColorFilters (Цветовые фильтры)

Справочник цветовых фильтров

Поле	Тип	Описание
DisplayName	string	Отображаемое имя
DocType_Id	string	Идентификатор документа
DocTypeName	string	Наименование документа
Script	string	Текст скрипта

Eos.DP.Wrapper.DocumentStatus (Статус документа)

Справочник статусов документа

Поле	Тип	Описание
Name	string	Наименование
EndFlag	bool	Финальный статус
Groups	Eos.DP.Wrapper.Status_Groups[]	Группы документов
DemandComment	bool	Обязателен комментарий
IsPositive	bool	Положительный
DocResponsibilityReturnFlag	bool	Возврат ответственности по документу

Eos.DP.Wrapper.ReportForm (Печатная форма)

Справочник печатных форм

Поле	Тип	Описание
FormName	string	Имя формы
FormData	Byte[]	Данные формы
ObjectTypeId	string	Тип объекта

Eos.DP.Wrapper.PrintFormFormat (Формат печатной формы)

Справочник форматов печатных форм

Поле	Тип	Описание
Name	string	Наименование
FileExtension	string	Расширение файла

Eos.DP.Wrapper.v2.5.1.DateRange (Диапазоны дат)

Справочник диапазонов дат

Поле	Тип	Описание
Name	string	Название диапазона
EndDate	int	Срок

Eos.DP.Wrapper.TaskTypeProcessing (Тип обработки поручения)

Справочник типов обработки поручения

Поле	Тип	Описание
Name	string	Наименование
AllowSubFolders	bool	Возможны вложенные папки

Eos.DP.Wrapper.BarCodeType (Тип штрихкода)

Справочник типов штрихкода

Поле	Тип	Описание
Name	string	Название

Eos.DP.Wrapper.WayType (Тип маршрута)

Справочник типов маршрута

Поле	Тип	Описание
Name	String	Название

Eos.DP.Wrapper.Docflow (Маршрут документа)

Справочник маршрутов документа

Поле	Тип	Описание
Name	string	Название маршрута
WorkflowType	WorkflowType	Тип процесса
IsDraft	bool	Это черновик
DocumentGroups	Eos.DP.Wrapper.DocFlowDocumentGroup[]	Группы документов
WayStages	Eos.DP.Wrapper.WayStage[]	Этапы маршрута
WayVariables	Eos.DP.Wrapper.WayVariable[]	Переменные

Eos.DP.Wrapper.WorkflowType (Тип процесса)

Справочник типов процесса

Поле	Тип	Описание
WorkflowName	string	Отображаемое имя процесса
RelativeUri	string	Относительный путь к файлу процесса
WorkflowHost	Eos.DP.Wrapper. WorkflowType_WfHost[]	Адрес хоста процессов
Config	string	Конфигурация
ConfigUserGroup_Id	Guid	Идентификатор группы пользователей-конфигураторов
ConfigUserGroupName	string	Наименование группы пользователей-конфигураторов
UserGroup_Id	Guid	Идентификатор группы пользователей
UserGroupName	string	Наименование группы пользователей

Eos.DP.Wrapper.WayStageType (Тип этапа маршрута)

Справочник типов этапа маршрута

Поле	Тип	Описание
Name	String	Название типа

Eos.DP.Wrapper.VarType (Тип переменной)

Справочник типов переменной (маршрута документа)

Поле	Тип	Описание
Name	String	Название

Eos.DP.Wrapper.FolderSMS (Папка CMP)

Справочник папок CMP

Поле	Тип	Описание
Name	String	Наименование
IsTaskTypeProcessingRoot	bool	Корневая для типа обработки
ProcessTypes	Eos.DP.Wrapper. FolderSMS_ProcessType[]	Типы обработки поручений

Секция ProcessTypes (типы обработки поручений):

TaskTypeProcessing_Id	Guid	Ссылка на тип обработки поручения
-----------------------	------	-----------------------------------

Справочники с рубрикацией

Будут приведены лишь некоторые такие справочники.

Eos.DP.Wrapper.RubricatedCatalog (Справочник с рубрикацией)

Предок всех пользовательских справочников системы.

Содержит одну секцию `Rubrics`, являющуюся коллекцией записей следующего вида:

Поле	Тип	Описание
<code>Rubric_Id</code>	<code>Guid</code>	Идентификатор рубрики (указатель на запись справочника <code>Eos.DP.Wrapper.Rubric</code>)
<code>RubricOrderNum</code>	<code>Int</code>	Порядковый номер (вес) записи справочника в этой рубрике

Eos.DP.Wrapper.Country (Страна)

Справочник видов стран. Потомок справочника с рубрикацией.

Поле	Тип	Описание
<code>Code</code>	<code>Int</code>	Код страны
<code>Name</code>	<code>String</code>	Название страны

Eos.DP.Wrapper.DocumentGroup (Группа документов)

Справочник групп документов. Потомок справочника с рубрикацией.

Поле	Тип	Описание
<code>Name</code>	<code>string</code>	Название группы документов
<code>DocTypeId</code>	<code>string</code>	Код типа документа для данной группы (получается константой вида <code><тип документа>.TypeId</code>)
<code>DocTypeName</code>	<code>string</code>	Строковое имя типа документа
<code>Numerator_Id</code>	<code>Guid</code>	Ссылка на справочник Нумератор
<code>DocNumberPrefix</code>	<code>string</code>	Префикс номера документа
<code>DocNumberSuffix</code>	<code>string</code>	Суффикс номера документа
<code>MakeRegistrarTheDocAuthor</code>	<code>bool</code>	Опция "Регистратор – автор документа"
<code>NotificationUrl</code>	<code>string</code>	Адрес оповещения по HTTP
<code>NotificationTimeout</code>	<code>int</code>	Макс. время ожидания оповещения по HTTP, сек
<code>NotifyCreateDocument</code>	<code>bool</code>	Оповещать при создании документа
<code>NotifyEditDocument</code>	<code>bool</code>	Оповещать при редактировании документа

NotifyCreateVersion	bool	Оповещать при создании версии документа
NotifyDeleteVersion	bool	Оповещать при удалении версии документа
NotifyDeleteDocument	bool	Оповещать при удалении документа
NotifyChangeAccess	bool	Оповещать при изменении списка доступа
NotifyReadDocument	bool	Оповещать при чтении документа
NotifyPrintDocument	bool	Оповещать при печати карточки документа
NotifyAddFile	bool	Оповещать при добавлении файла
NotifyDeleteFile	bool	Оповещать при удалении файла
NotifyReadFile	bool	Оповещать при чтении файла
NotifyRenameFile	bool	Оповещать при переименовании файла
NotifyEditFile	bool	Оповещать при редактировании файла
NotifyDiscuss	bool	Оповещать при согласовании
NotifyTransfer	bool	Оповещать при передаче
NotifyLogicalDelete	bool	Оповещать при логическом удалении
NotifyLogicalRestore	bool	Оповещать при восстановлении документа
EmailSubject	string	Шаблон темы оповещения
EmailBody	string	Шаблон текста оповещения
UniqueType_Id	Guid	Идентификатор уникального типа
RegisterOnlyWithParent	bool	Создавать как связанный
ImmediateBarCoding	bool	Печать штрихкода после создания
GroupComment	string	Комментарий для группы
DEFAULTVALUESSCRIPT	string	Скрипт дефолтовых значений
BEFORESAVESCRIPT	string	Скрипт перед сохранением
AFTERSAVESCRIPT	string	Скрипт после сохранения
PrintFormFormat_Id	Guid	Ссылка на справочник «Формат печатной формы»
AllowAutoregistration	bool	Разрешена авторегистрация
NotStrictEditRules	bool	Разрешить изменение основных атрибутов поручения после прочтения

		исполнителем
SMEV SID	string	Идентификатор сервиса СМЭВ
SMEV Method	string	Идентификатор метода сервиса СМЭВ
SMEV ResponseTemplate	string	Шаблон форматирования ответов из СМЭВ
PGU Request	string	Идентификатор госуслуги
PGU RequestTemplate	string	Шаблон форматирования запроса с ПГУ
Prints	Eos.DP.Wrapper. DocGroup_Print[]	Доступные печатные формы
RepeatabilityCheck	bool	Проверять на повторность
IsPrintAfterCreate	bool	Печать после создания
CreateReportText	string	Подпись кнопки создания отчета
AttachPrintForm	bool	Прикреплять печатную форму при отправке адресатам
AttachDocumentPassport	bool	Прикреплять паспорт документа при отправке адресатам
ScreenFormIdOnCreate	Guid	Экранная форма при создании
ScreenFormIdOnView	Guid	Экранная форма при просмотре
ScreenFormIdOnPreview	Guid	Экранная форма при предпросмотре
ScreenFormIdOnEdit	Guid	Экранная форма при редактировании
DuplicatesSearchSettings_Id	Guid	Ссылка на поисковый запрос для проверки на повторность
IsDocumentItem	bool	Данная группа представляет пункт документа
EndAllChildTasks	bool	При закрытии - закрывать все подчиненные поручения
EndTaskIfAllReportsEnded	bool	Закрывать поручение, если отчеты всех исполнителей имеют финальный статус
EndTaskIfRespReportEnded	bool	Закрывать поручение, если отчет ответственного исполнителя имеет финальный статус
EndSubLevelTasks	bool	При закрытии - закрывать подчиненные поручения на 1 уровень вниз
EndDocOnEndTaskByResp	bool	Закрывать документ / пункт при закрытии поручения ответственными по документу
EndDocOnEndAllTaskInGroup	bool	Закрывать документ / пункт при закрытии всех поручений данной

		группы
EndTasksUnderControl	bool	Закрывать контрольные
BarCodeType_Id	Guid	Стандарт печати штрихкода. Ссылка на справочник «Тип штрихкода»
GenerateNumberOnSave	bool	Генерация номера документов при сохранении
UsedInResponsibilityLogic	bool	Участвует в определении ответственного по документу
EndTaskFirstPositiveReport	bool	Закрывать поручение, если отчет исполнителя имеет финальный положительный статус
EndTaskFirstNonPositiveReport	bool	Закрывать поручение, если отчет исполнителя имеет финальный неположительный статус
FolderSMS_Id	Guid	Ссылка на папку СМР

Справочник также содержит следующие секции:

Секция CopyScripts содержит информацию о скриптах копирования:

Name	string	Название
Script	string	Текст скрипта

Секция LinkedGroups содержит информацию о группах документов, документы из которых могут быть созданы, как связанные:

LinkType_Id	Guid	Идентификатор типа ссылки по умолчанию
ParentDocumentGroup_Id	Guid	Идентификатор группы документов

Справочник групп документов содержит также секцию DefaultACL (Список доступа по умолчанию). Её заполнение через API допустимо только пользователями с ролью Системный администратор. Состав этой секции отражает состав списка доступа документа (описанного в разделе "Документы")

Eos.DP.Wrapper.DocumentSearchSettings (Настройки поиска документов)

Справочник поисков документов, в том числе фильтров. Потомок справочника с рубрикацией.

Поле	Тип	Описание
DisplayName	string	Название поискового запроса
FormData	string	Схема формы
Owner_Id	Guid	Владелец поискового запроса, ссылка на субъект безопасности

ImmediatelyRun	bool	Выполнять сразу
MarkUnreaded	bool	Выделять непрочитанные
Folder_Id	Guid	Идентификатор папки или рубрики, если это фильтр
ColorFilters	Eos.DP.Wrapper. DocSearchSettings_ColorFilter[]	Цвета
PrintForm	byte[]	Печатная форма для поиска
PrintFormFormat_Id	Guid	Ссылка на формат печатной формы
PrintFormName	string	Имя печатной формы
OpenDocumentInsteadTask	bool	Открывать документ вместо поручения
IsTableView	bool	Табличное представление результатов
IsPreviewEnabled	bool	Предпросмотр документа
EmptyCriteriesWarning	bool	Вкл. подтверждение запуска без критериев

Eos.DP.Wrapper.Folder (личная папка)

Справочник личных папок пользователей. Приведены некоторые поля.

Поле	Тип	Описание
Owner_id	Guid	Владелец
ShowUnreaded	bool	Показывать непрочтенные
ShowBolded	bool	Выделять жирным
IsShared	bool	Разрешить общий доступ к папке
InheritACLFromParent	bool	Наследовать права от родительской папки. Если Да , список доступа папки и реквизиты Разрешить общий доступ к папке, Предоставлять доступ к документам.. имеют значения, равные родительской папки.
InheritACLToChild	bool	Наследовать права дочерним папкам. Если Да , при изменении списка доступа данной папки Система для всех папок в дереве ниже, у кот Наследовать права

		от родительской папки = Да , устанавливает идентичный список доступа и реквизиты Разрешить общий доступ к папке, Предоставлять доступ...
GiveAccessAfterPuttingDoc	bool	Предоставлять доступ к документам при помещении документа в папку
AutoRefresh	bool	Автоматическое обновление
RefreshPeriod	int	Период автоматического обновления, мин
IsInboxFolder	bool	Папка для входящих документов
IsOutboxFolder	bool	Папка для исходящих документов
IsRecentFolder	bool	Папка для недавно открытых документов
IsControlFolder	bool	Папка для контрольных документов
FolderNearNumber	Guid	Число с папкой. Определяет, что показывать в скобках рядом с названием личной папки: () ничего (*) количество непрочитанных документов (по умолчанию) () количество всех документов () количество непрочитанных / всех документов Значения 0 и 0/0 опускаются, т.е. в этих случаях ничего не выводится.
HideNameForSingleFilter	bool	Скрывать название единственного фильтра
AcceptAllGroupsToInbox	bool	Все группы поручений
AcceptAllGroupsToOutbox	bool	Все группы поручений

Справочники "Контакт"

Справочник "Контакт" является, на самом деле, группой из пяти справочников: базового и четырёх порожденных справочников: Гражданин, Организация, Должностное лицо и Подразделение. Базовый справочник имеет общую таблицу для всех потомков, что позволяет ссылаться из других типов системы на "контакт вообще". Реальный тип записи хранится в унаследованном поле `_TypeId`.

Eos.DP.Wrapper.Uoi.UoiBase (Контакт)

Предок четырёх справочников, образующих справочник "Контакт". Допустимо загружать записи такого типа, используя метод `LoadCatalog` и пользоваться только базовыми реквизитами, но сохранять можно только объекты одного из четырёх порожденных типов.

Поле	Тип	Описание
DisplayName	string	Отображаемое имя
Code	string	Индекс
Phone	string	Телефон
Email	string	Адрес электронной почты
MEDO_Identifier	Guid	Идентификатор МЭДО
SearchPriority	int	Приоритет при поиске

Eos.DP.Wrapper.Uoi.Citizen(Гражданин)

Потомок справочника "Контакт". Доступен для дополнительной настройки в Конфигураторе.

Поле	Тип	Описание
LastName	string	Фамилия
FirstName	string	Имя
FathersName	string	Отчество
BirthDate	DateTime	Дата рождения
AddressType_Id	Guid	Идентификатор типа адреса (ссылка на справочник Eos.DP.Wrapper.AddressType)
Street	string	Улица
Town	string	Город
Region	string	Область, край
Country_Id	Guid	Идентификатор страны (ссылка на справочник Eos.DP.Wrapper.Country)
IdentityType_Id	Guid	Идентификатор удостоверения личности (ссылка на справочник Eos.DP.Wrapper.IdentityType)
Serial	string	Серия
Number	string	Номер
GivenDate	DateTime	Дата выдачи документа
GivenBy	string	Кем выдан документ
INN	string	ИНН

WebSite	string	Сайт
SNILS	string	СНИЛС
OtherInfo	string	Дополнительная информация

Eos.DP.Wrapper.UoI.Organization (Организация)

Потомок справочника "Контакт". Доступен для дополнительной настройки в Конфигураторе.

Поле	Тип	Описание
Name	string	Наименование полное
ShortName	string	Наименование краткое
AddressType_Id	Guid	Идентификатор типа адреса (ссылка на справочник Eos.DP.Wrapper.AddressType)
Street	string	Улица
Town	string	Город
Region	string	Область, край
Country_Id	Guid	Идентификатор страны (ссылка на справочник Eos.DP.Wrapper.Country)
WebSite	string	Сайт
Fax	string	Факс
INN	string	ИНН
KPP	string	КПП
OKPO	string	ОКПО
OKONH	string	ОКОНХ
OKVED	string	ОКВЭД
OGRN	string	ОГРН
Account	string	Р. счет №
CorrespondingAccount	string	Кор. счет №
BankName	string	Название банка
BankAddress	string	Адрес банка
BankBIC	string	БИК банка
MEDO_Source_UID	string	Идентификатор в МЭДО
MEDO_Address	string	Адрес шлюза МЭДО

MEDO_1	string	Значения для МЭДО
--------	--------	-------------------

Eos.DP.Wrapper.Uoi.Official (Должностное лицо)

Потомок справочника "Контакт". Доступен для дополнительной настройки в Конфигураторе.

Поле	Тип	Описание
LastName	string	Фамилия
FirstName	string	Имя
FathersName	string	Отчество
PostName	string	Должность
IsChief	bool	Признак "Это руководитель"
Room	string	Комната
WebSite	string	Web-страница
IsInboxRegistrator	bool	Регистратор входящей корреспонденции
Snils	string	СНИЛС

Секция "Рубрики" справочника "Должностное лицо", как и её предок, содержит ссылку `Rubric_Id`. Однако это поле ссылается не на справочник рубрик, а на справочник Контакт. Должностные лица могут входить только в организации и подразделения.

Eos.DP.Wrapper.Uoi.Department (Подразделение)

Потомок справочника "Контакт". Доступен для дополнительной настройки в Конфигураторе.

Поле	Тип	Описание
Name	string	Наименование

Секция "Рубрики" справочника "Подразделение", как и её предок, содержит ссылку `Rubric_Id`. Однако это поле ссылается не на справочник рубрик, а на справочник Контакт. Подразделения могут входить только в организации и другие подразделения.

Документы

Данный раздел описывает объекты, соответствующие документам системы.

Eos.DP.Wrapper.Doc.BaseDocument (Документ)

Базовый документ. Все документы порождаются либо от BaseDocument, либо от его потомков и содержат все описанные в нём реквизиты.

Потомок класса Eos.DP.Wrapper.Head (Головной секции). Содержит дополнительно одно поле DocGroup_Id, указывающее на группу документов, и ряд секций.

Поле	Тип	Описание
DocGroup_Id	Guid	Идентификатор группы документов (указатель на справочник групп документов)
DocGroupName	string	Название группы документов (поле расширенной ссылки)

Содержимое **секции ACL** описывает права доступа к данному документу:

Поле	Тип	Описание
SecuritySubject_Id	Guid	Идентификатор субъекта безопасности (пользователя или группы пользователей)
CanRead	bool	Право на чтение
CanReadFiles	bool	Право на чтение файлов
CanEdit	bool	Право на редактирование
CanAddFiles	bool	Право на добавление файлов
CanEditFiles	bool	Право на редактирование файлов
CanDeleteFiles	bool	Право на удаление файлов
CanDelete	bool	Право на удаление документа
CanCreateVersion	bool	Право на управление версиями документа
CanChangeAccess	bool	Право на управление доступом
CanDiscuss	bool	Право на согласование
CanTransfer	bool	Право на передачу

Секция Folders содержит реквизиты личных папок, в которых находится документ:

Поле	Тип	Описание
Folder_Id	Guid	Идентификатор личной папки пользователя

Секция Versions содержит основные реквизиты и секции всех документов и расширяется в порожденных классах. Добавляя в конфигураторе поля в документ, вы фактически добавляете их в секцию Versions.

Состав секции Versions в базовом документе:

Поле	Тип	Описание
VersionNumber	int	Целочисленный номер версии
IsActual	bool	Признак актуальности (установлен только у одной версии документа)
Date	DateTime	Дата регистрации документа
Number	string	Номер документа
ANNOTATION	string	Аннотация
BeginDate	DateTime	Дата начала выполнения
DelayDays	DateTime	Нарушение срока
EndDate	DateTime	Срок
EndRealDate	DateTime	Дата выполнения

Кроме того, секция Versions содержит несколько секций, содержащих групповые реквизиты: Рубрики (Rubrics), Файлы (Files), Авторы (Authors), Ссылки (Links), Поручения (Tasks), Обсуждение (Discussion), Ответственные по поручениям (Responsibles) и Первые получатели (FirstReceivers).

Содержимое секции Rubrics:

Поле	Тип	Описание
Rubric_Id	Guid	Идентификатор рубрики (указатель на справочник рубрик документов)

Содержимое секции Files:

Поле	Тип	Описание
File_Id	Guid	Идентификатор бинарного блока (закачанного командой SaveSessionData) или сохранённого в системе файла
FileName	string	Имя файла
FileSize	decimal	Размер файла в байтах

Содержимое секции Authors:

Поле	Тип	Описание
Author_Id	Guid	Идентификатор автора (указатель на справочник Контакт)

Содержимое секции Links:

Поле	Тип	Описание
Doc_Id	Guid	Идентификатор документа, на который установлена ссылка
DocNumber	string	Номер ссылаемого документа
DocDate	DateTime	Дата ссылаемого документа
DocGroupName	string	Название группы ссылаемого документа

Содержимое секции Tasks:

Поле	Тип	Описание
Author_Id	Guid	Идентификатор автора поручения
AuthorDisplayName	string	Отображаемое имя автора
Date	DateTime	Дата поручения
TaskGroupId	string	Название группы поручения
Task_Id	Guid	Идентификатор поручения
TaskUrgencyId	Guid	Идентификатор срочности поручения
TaskUrgencyName	string	Наименование срочности поручения
Text	string	Текст поручения
TypeName	string	Тип маршрута
WayTypeId	Guid	Идентификатор типа маршрута
IsProject	bool	Поручение является черновиком
TaskEndDate	DateTime	Дата закрытия поручения
TaskController_Id	Guid	Контроллер поручения
TaskNumber	String	Номер поручения
TaskOrder	string	Порядок поручения
TaskTypeId	string	Тип поручения
IsHidden	bool	Невидимое
TaskIntOrder	int	Порядок поручения (в числовом виде)
ParentTaskId	Guid	Родительское поручение
MultiDocumentGuid	Guid	Флаг мультисоздания
ProjectPartiallyAccepted	bool	Утверждено не полностью

EXECUTIONPROCESS	string	Ход исполнения
ClosingDocNumber	string	Номер закрывающего документа
ControlCategory_Id	Guid	Категория контроля (ссылка на справочник ControlCategory)
ControlCategoryIsUnderControl	bool	Подлежит контролю (флаг дополняет ссылку на категорию контроля)
ResponsibleDepartament	string	Ответственный департамент
ExternalControlId	Guid	Внешний контроль (идентификатор документа)
ExternalControlNumber	string	Внешний контроль (номер документа)
ExternalControlDate	DateTime	Внешний контроль (дата документа)

Секция Tasks содержит секцию Reports (отчёты) :

Поле	Тип	Описание
Performer_Id	Guid	Идентификатор исполнителя
PerformerDisplayName	string	Отображаемое имя исполнителя
ReadDate	DateTime	Дата прочтения
ReceiveDate	DateTime	Дата получения
ReportId	Guid	Идентификатор отчёта
ReportStatusId	Guid	Идентификатор состояния выполнения
ReportStatusEndFlag	bool	Флаг завершения отчёта
ReportStatusName	string	Состояние выполнения
SaveDate	DateTime	Дата сохранения
SampleType_Id	Guid	Тип экземпляра документа (ссылка на справочник DocumentSampleType)
EndDate	DateTime	Персональный срок

Содержимое секции Discussion:

Поле	Тип	Описание
MessageType_Id	Guid	Ссылка на тип сообщения (справочник MessageType)
Author_Id	Guid	Контакт (ссылка на справочник UolBase)
COMMENT	string	Текст сообщения

Содержимое секции Responsibles:

Поле	Тип	Описание
ResponsibleDepartment_Id	Guid	Ответственный департамент (ссылка на UolBase)
ResponsiblePerformer_Id	Guid	Ответственный исполнитель (ссылка на UolBase)
ResponsibleTaskId	Guid	Идентификатор поручения
RootTaskId	Guid	Идентификатор первого поручения
RootEndDate	DateTime	Срок первого поручения
RootEndRealDate	Datetime	Фактическая дата первого поручения
RootControlCategory_Id	Guid	Категория контроля (ссылка на ControlCategory)
RootDocumentItem	string	Пункт первого поручения
RootTaskBeginDate	DateTime	Дата начала выполнения
RootDocumentStatus_Id	Guid	Статус документа (ссылка на DocumentStatus)

Содержимое секции FirstReceivers:

Поле	Тип	Описание
Receiver_Id	Guid	Получатель (ссылка на справочник UolBase)
IsResponsible	bool	Является ли ответственным

Eos.DP.Wrapper.Doc.Task (Поручение)

Потомок класса Eos.DP.Wrapper.BaseDocument (Базовый документ). Содержит дополнительно ряд полей. В секции Versions:

Поле	Тип	Описание
TaskUrgency_Id	Guid	Срочность поручения
TaskUrgencyName	string	Название срочности поручения
IsConfidential	bool	Конфиденциально
WayType_Id	Guid	Тип маршрута
WayTypeName	string	Название типа маршрута
IsProject	bool	Является поручение проектом или нет
Perfomers	Eos.DP.Wrapper.Task_Version_Perfomer[]	Исполнители
DocumentItem	string	Пункт документа
ResponsibleDepartment_Id	Guid	Ответственное подразделение (ссылка на справочник Department)

DICONTENT	string	Содержимое пункта документа
ResponsiblePerformer_Id	Guid	Ответственный исполнитель (ссылка на справочник UolBase)

Секция `Performers` описывает исполнителей поручения:

Поле	Тип	Описание
Performer_Id	Guid	Исполнитель
PerformerDisplayName	string	Отображаемое имя исполнителя
ReceiveDate	DateTime	Дата получения
BeginDate	DateTime	Дата начала исполнения
TaskViewDate	DateTime	Дата просмотра поручения
EndDate	DateTime	Срок
IsResponsible	bool	Ответственный исполнитель
CanEdit	bool	Редактирование
CanCreateVersion	bool	Создание версии
CanDelete	bool	Удаление
CanDiscuss	bool	Согласование
CanAddFiles	bool	Добавление файлов
CanEditFiles	bool	Редактирование файлов
CanDeleteFiles	bool	Удаление файлов
CanChangeAccess	bool	Управление доступом
CanReadFiles	bool	Чтение файлов
CanTransfer	bool	Передача документа

Флаги `CanEdit`, `CanCreateVersion` и т.д. являются правами исполнителя по отношению к документу, по которому выдано поручение.

Eos.DP.Wrapper.Doc.Report (Отчёт)

Потомок класса `Eos.DP.Wrapper.BaseDocument` (Базовый документ). Содержит дополнительно ряд полей.

В секции `Versions`:

Поле	Тип	Описание
StatusDate	DateTime	Дата установления статуса
PerfomerRowId	Guid	Идентификатор строки автора отчета в

		секции "Исполнители" в типе "Поручение"
Responsibles	CSPILH7RRAG2GERJCAWK3SH[]	Ответственные по поручениям
FirstReceivers	CSW6N32CY26KXUPBNTDR54E[]	Первые получатели

Eos.DP.Wrapper.Doc.OfficialDocument (Служебный документ) и его наследники

Потомок класса `Eos.DP.Wrapper.BaseDocument` (Базовый документ). От него наследуются такие типы документов, как Входящий документ (`IncomingDocument`), Исходящий документ (`OutgoingDocument`), Заявка с ПГУ (`PGURequest`), Обращение гражданина (`CitizensAppeal`), Внутренний документ (`InternalDocument`), Документ Multimedia (`MultimediaDocument`), Проект (`Project`), Ответ на обращение гражданина (`CitizensAppealAnswer`), Регистр (`Eos.DP.Wrapper.Registry`), Уведомление о входящем МЭДО (`Eos.DP.Wrapper.MEDONotification`). Пространство имен упомянутых типов документов, для которых оно не указано: `Eos.DP.UserDefinedTypes`.

Секция `Versions` служебного документа включает:

Поле	Тип	Описание
Case_Id	Guid	Списан в дело (ссылка на справочник Номенклатура дел <code>Nomenclature</code>)
CaseNumber	string	Индекс дела (дополняет поле <code>Case_Id</code>)
CaseName	string	Заголовок дела (дополняет поле <code>Case_Id</code>)
CaseTransferDate	DateTime	Дата списания в дело
DocTransfer	<code>Eos.DP.Wrapper.Doc.OfDoc_Transfer[]</code>	Журнал передачи документа
Addressees	<code>Eos.DP.Wrapper.Doc.OfDoc_Addressees[]</code>	Адресаты
Vise	<code>Eos.DP.Wrapper.Doc.OfDoc_Vise[]</code>	Визы
Executors	<code>Eos.DP.Wrapper.Doc.OfDoc_Executors[]</code>	Исполнители
DocKind0_Id	Guid	Вид документа (ссылка на справочник <code>DocKind</code>)
CaseTransferer_Id	Guid	Кто списал в дело (ссылка на справочник <code>Official</code>)
Secrecy0_Id	Guid	Гриф доступа (ссылка на справочник <code>Secrecy</code>)
DocRecType0_Id	Guid	Способ получения документа (ссылка на справочник <code>DocRecType</code>)

Responsibles	Eos.DP.UserDefinedTypes. CSHK7Z26CV352EJASLWQPEA[]	Ответственные по поручениям
CasePart	string	Часть дела
OriginalReturned	bool	Подлинник возвращен исполнителем

Приложение 2. Синтаксис поисковых выражений метода SearchCatalog

Метод SearchCatalog принимает в качестве третьего параметра строку, содержащую поисковый запрос, удовлетворяющий определенному синтаксису.

В общем случае строка поискового запроса может быть описана в таком виде:

```
<список возвращаемых полей>:<список критериев>
```

, т.е., два списка разделяются символом ":".

<список возвращаемых полей> - набор реквизитов типа справочников, поиск которых мы производим, т.е.

```
<список возвращаемых полей> := <название реквизита>[,<название реквизита>...]
```

Под названием реквизита типа понимается API имя этого реквизита, например "Name". Существует также возможность запрашивать реквизиты дочерних секций типа справочника, а также, реквизитов справочников, на которые данный справочник имеет ссылки. В случае с дочерними секциями название реквизита является составным, вида «API имя секции». «API имя реквизита». В случае со ссылочными реквизитами название реквизита является также составным, вида «API имя ссылки». «API имя реквизита ссылочного справочника»

Пример списка возвращаемых полей:

```
_Id,Name,DisplayName,Section.Field,Link.Field,Section.Link.Field
```

<список критериев> - набор критериев, по которым может производиться поиск элементов справочника заданного типа.

```
<список критериев> := <операция>(<первый операнд>,<второй операнд>)  
[<операция>(<первый операнд>,<второй операнд>) ...]
```

Таким образом, в описании критериев используется польская запись выражения: сначала указывается операция, а затем операнды.

<операция> может принимать ряд predefined значений, а именно:

Операция	Описание	Ограничения
=	Равенство	
!=	Неравенство	
>	Операнд 1 больше операнда 2	
>=	Операнд 1 больше либо равен операнду 2	
<	Операнд 1 меньше операнда 2	
<=	Операнд 1 меньше либо равен операнду 2	
like	Сравнение строк (аналогичен оператору LIKE SQL)	Операнд 1 – строковый реквизит, операнд 2 – строковый параметр
tq	Запрос к текстовому полю (см.	Операнд 1 – строковый

	«Краткое руководство пользователя»)	реквизит, операнд 2 – строковый параметр
isnull	Операнд 1 является NULL в БД	
contains	Операнд 1 содержит операнд 2, полнотекстовый поиск.	
in	Операнд 1 содержится в массиве значений, представленном операндом 2	Операнд 2 должен быть представлен именно массивом (не списком)
inids	Операнд 1 содержится в массиве значений, представленном операндом 2 в виде строки, содержащей значения, которые разделены вертикальной чертой .	

<первый операнд> - название реквизита (см. Приложение 1).

<второй операнд> - символ «?», если второй операнд будет задаваться внешним параметром и строка “field, catalog.<название реквизита>”, если второй операнд – другой реквизит этого же справочника.

Пример критериев:

```
=(_Id,?)!=(NumberOne,?)<(NumberOne,field,catalog.NumberTwo)
```

Пробелы не допустимы.

Примеры

Поиск имени группы документов по уникальному индексу:

```
DataSet ds = systemAPI.SearchCatalog(token, DocumentGroup.TypeId,
    "Name:=(_Id,?)", groupId);
string groupName = ds.Tables[0].Rows[0]["Name"].ToString();
```

Поиск подразделений организации:

```
DataSet ds = systemAPI.SearchCatalog(token, Department.TypeId,
    "_Id,Name:=(Rubrics.Rubric_Id,?)", organizationId);
List<Guid> departments = new List<Guid>();
foreach (DataRow row in ds.Tables[0].Rows)
    departments.Add((Guid)row["_Id"]);
```

Приложение 3. Задание критериев поиска документов. Объект CriteryGroup

Метод `SearchDocsByCriteries` позволяет искать документы системы, удовлетворяющие определенным критериям.

Третьим параметром команды является экземпляр класса `CriteryGroup`, содержащий критерии либо вложенные группы критериев, по которым будет производиться выборка документов.

Метод возвращает объект `DataSet`, содержащий таблицу "Root" следующего содержания:

<code>_Id:</code>	идентификатор документа
<code>Date:</code>	дата документа
<code>EditDate:</code>	дата последнего редактирования документа

Пример:

```
CriteryGroup cg = new CriteryGroup(GroupType.Or);
cg.AddFieldCritery("doc.Versions.Date", CriteryType.Equal,
    "doc.Versions.Number");
cg.AddTextCritery("doc.Versions.Number", "123 321", false);
DataSet ds = api.SearchDocsByCriteries(token, BaseDocument.TypeId, cg);
foreach (DataRow row in ds.Tables[0].Rows)
{
    DataSet documentBuffer = api.LoadDocument(token, (Guid)row["_Id"]);
    var document = new BaseDocument(documentBuffer);
    // Здесь выполняются действия над найденным и загруженным документом
}
```

CriteryGroup

Этот класс предназначен для группировки критериев поиска. Критерии могут группироваться по условию «И» или «ИЛИ». Тип группировки определяется при создании экземпляра класса `CriteryGroup`.

```
CriteryGroup group = new CriteryGroup(GroupType.And);
```

Класс `CriteryGroup` содержит методы для добавления критериев различных типов.

Опишем методы `CriteryGroup`.

AddFieldCritery

```
AddFieldCritery(string fieldNameLeft, CriteryType criteryType,
    string fieldNameRight)
```

Этот метод добавляет критерий, который означает, что первый реквизит документа (`fieldNameLeft`) находится в каком-то отношении со вторым реквизитом (`fieldNameRight`).

Тип отношения определяется перечислением `CriteryType`. Допустимыми являются следующие значения:

CriteryType	Описание
Equal	равенство
NotEqual	неравенство

Like	
Greater	Операнд 1 больше операнда 2
GreaterOrEqual	Операнд 1 больше либо равен операнду 2
Less	Операнд 1 меньше операнда 2
LessOrEqual	Операнд 1 меньше либо равен операнду 2

Операнды (`fieldNameLeft` и `fieldNameRight`) задаются в виде строк следующим образом: `"doc.<название реквизита типа документов>"`. Под названием реквизита типа документов понимается API имя этого реквизита, например, `"Name"` (задается в Конфигураторе, подробнее см. «Краткое руководство пользователя»). Существует также возможность запрашивать реквизиты дочерних секций и реквизитов справочников, на которые данный документ имеет ссылки, подробнее см. Приложение 2. Префикс `«doc.»` в случае с дочерними секциями и ссылочными реквизитами также необходим.

AddFullTextCriteriy

```
AddFullTextCriteriy(string fieldName, string value)
```

Этот метод добавляет критерий полнотекстового поиска, который означает, что строка `value` должна иметь вхождения в прикрепленные к документу файлы. Допустимые значения параметра `fieldname` включают:
`"doc.Versions.Files.File.CONTENT"`, `"FULLTEXTSEARCHSTRING"`.

AddGroupCriteriy

```
AddGroupCriteriy(GroupType type)
```

Этот метод добавляет группу критериев, содержащую список дочерних критериев или групп и объединяющую их по условию, зависящему от перечисления `GroupType`.

Метод возвращает экземпляр класса `GroupCriteriy`.

С помощью этого метода можно строить довольно сложные иерархии критериев.

Пример:

```
CriteriyGroup group1 = new CriteriyGroup(GroupType.Or);
group1.AddFieldCriteriy("doc.Versions.ANNOTATION", CriteriyType.Equal,
    "doc.Versions.Number");
CriteriyGroup group2 = group1.AddGroupCriteriy(GroupType.And);
group2.AddFullTextCriteriy("doc.Versions.Files.File.CONTENT",
    "Столица", false);
group2.AddFieldCriteriy("doc.Versions.Number", CriteriyType.Less,
    "doc.Versions.VersionNumber");
```

AddNullCriteriy

```
AddNullCriteriy(string fieldName)
```

Этот метод добавляет критерий, который означает, что реквизит документа равен NULL в БД.

Пример:

```
CriteryGroup group1 = new CriteryGroup(GroupType.Or);  
group1.AddNullCritery("doc.Versions.Date");
```

AddNotNullCritery

```
AddNotNullCritery(string fieldName)
```

Этот метод добавляет критерий, который означает, что реквизит документа не равен NULL в БД.

Пример:

```
CriteryGroup group1 = new CriteryGroup(GroupType.Or);  
group1.AddNotNullCritery("doc.Versions.Date");
```

AddParamCritery

```
AddParamCritery(string fieldName, CriteryType criteryType, DbType type,  
object value)
```

```
AddParamCritery(string fieldName, CriteryType criteryType, DbType type, Array  
values)
```

Этот метод добавляет критерий, который означает, что реквизит документа должен находиться в каком-то отношении со значением value или values. Тип отношения определяется перечислением CriteryType. Допустимы следующие значения:

Операция	Описание
Equal	равенство
NotEqual	неравенство
Greater	Операнд 1 больше операнда 2
GreaterOrEqual	Операнд 1 больше либо равен операнду 2
Less	Операнд 1 меньше операнда 2
LessOrEqual	Операнд 1 меньше либо равен операнду 2
Like	Сравнение строк (аналогичен оператору LIKE SQL)
TextQuery	Строка поиска по текстовым полям (см. Краткое руководство пользователя)
Contains	Строка полнотекстового поиска БД
In	Аналогичен оператору «in» SQL

При вызове этого метода необходимо указывать тип значения value (values) в формате ADO.NET DbType.

Пример:

```
CriteriyGroup group1 = new CriteriyGroup(GroupType.Or);
group1.AddParamCriteriy("doc.Versions.Number",
    CriteriyType.Equal, DbType.String, "123");
```

AddTextCriteriy

```
AddTextCriteriy(string fieldName, string value, bool ignoreCase)
```

Этот метод добавляет критерий, который означает, что реквизит документа должен удовлетворять определенному запросу к текстовым полям. Подробнее о синтаксисе таких запросов можно почитать в «Кратком руководстве пользователя» системы.

Пример:

```
CriteriyGroup group1 = new CriteriyGroup(GroupType.Or);
group1.AddTextCriteriy("doc.Versions.Number", "123 321", false);
```

AddSubqueryCriteriyExists

```
CriteriyGroup AddSubqueryCriteriyExists(CriteriySubquery sub, bool caseMode)
CriteriyGroup AddSubqueryCriteriyExists(CriteriySubquery sub)
```

Этот метод добавляет критерий, который означает, что подзапрос возвращает хотя бы 1 строку (аналогично оператору SQL exists).

AddSubqueryCriteriyFrom

```
CriteriyGroup AddSubqueryCriteriyFrom(string fieldNameLeft, CriteriySubquery sub,
string aliasDotField)
```

Этот метод добавляет указанный подзапрос sub, связанный с запросом с помощью left join по указанным полям, о которых речь далее. Поле fieldNameLeft, – это реквизит документа, а aliasDotField – это произвольное имя указанного подзапроса и 1 из возвращаемых им столбцов, разделенные точкой.

AddSubqueryCriteriyIn

```
CriteriyGroup AddSubqueryCriteriyIn(string fieldNameLeft, CriteriySubquery sub)
```

Этот метод добавляет критерий, который означает, что реквизит документа fieldNameLeft должен находиться в результатах подзапроса sub.

Приложение 4. Скрипты.

Реализована возможность задавать скрипты для решения задач:

- Копирование информации с одного документа в другой при создании связанного документа (скрипт задается на группе документов)
- Инициализация дефолтовыми значениями документа при создании (скрипт задается на группе документов)
- Реализация пользовательской логики сохранения документа (скрипт задается на группе документов)
- Пометка цветом документов в списках, удовлетворяющих определенным условиям (скрипт задается в справочнике цветовые фильтры)

Скрипт – это код на языке C#, который выполняется в отдельном домене (т.е. не имеет доступа к функционалу сервера системы eDocLib и не может ему повредить). В коде можно оперировать типизированными обертками над документами, так же, как и при программировании через АПИ.

В разделе описано создание всех видов скриптов.

Скрипты значений по умолчанию

Скрипты, описанные в этой секции, выполняются при создании нового документа. Они позволяют автоматически заполнять указанные поля документа.

Доступные переменные:

`dest` – переменная имеет тип обёртки типа документа, который указан на группе документов, для которой создаётся скрипт. Т.е. если группа документов создана от типа «Документ» - тип переменной `dest` будет `Eos.DP.Wrapper.Doc.BaseDocument`, если от типа «Поручение», то тип `dest` – `Eos.DP.Wrapper.Doc.Task` и т.д.

`Catalogs` – переменная имеет тип `IScriptHelper`, она нужна для организации взаимодействия скрипта с eDocLib'ом.

Пример:

При создании документов группы «Поручение начальникам подразделений» нужно в секции «Исполнители» указать начальников подразделений и установить признак «Конфиденциально»:

```
var task = dest.Versions.Single(x=>x.IsActual);

// Устанавливаем признак «Конфиденциально» и вписываем начало текста
// поручения
task.IsConfidential = true;
task.Text = "Начальникам подразделений";

// Добавляем записи в секцию исполнителей, идентификаторы получаем при помощи
// методов Catalogs
var performer = task.AddPerfomersRow();
performer.Perfomer_Id = Catalogs.GetUolIdByName("Иванов");
```

```
performer = task.AddPerfomersRow();
performer.Performer_Id = Catalogs.GetUolIdByName("Петров");

// Пример удаления записей из секции
performer = task.Performers.First(x => x.Performer_Id ==
Catalogs.GetUolIdByName("Петров"));
performer.Delete();

// Устанавливаем права на чтение поручения руководителям подразделений
// Для получения идентификаторов секции ACL используем методы
GetUserIdByLogin и GetUserIdByName
var ACL = dest.AddACLRow();
ACL.SecuritySubject_Id = Catalogs.GetUserIdByLogin("Ivanov");
ACL.CanRead = true;

ACL = dest.AddACLRow();
ACL.SecuritySubject_Id = Catalogs.GetUserIdByName("Петров");
ACL.CanRead = true;
```

Скрипты копирования

Скрипты, описанные в этой секции выполняются при создании связанного документа. Они позволяют автоматически заполнять указанные поля документа на основе данных родительского документа.

Доступные переменные:

src – обёртка типа документа, от которого создаётся связанный документ.

actual – актуальная версия src.

dest – обёртка типа связанного документа.

Catalogs – переменная имеет тип IScriptHelper, она нужна для организации взаимодействия скрипта с eDocLib'ом.

Пример:

При создании связанного документа «Благодарность» от документа, нужно в текст благодарности внести отображаемые имена авторов документа, а авторами документа указать начальство:

```
var destVersion = dest.Versions[0];

var txt = "Выносятся благодарность в лице руководства за отличную работу ";

// Проходим по секции Authors документа и добавляем имена в переменную txt
foreach (var row in destVersion.Authors)
    txt += row.AuthorDisplayName + ", ";

// Указываем авторов документа
var authors = destVersion.AddAuthorsRow();
authors.Author_Id = Catalogs.GetUolIdByName("Иванов");
authors = destVersion.AddAuthorsRow();
authors.Author_Id = Catalogs.GetUolIdByName("Петров");

// Заполняем поле Аннотация текстом благодарности
destVersion.ANNOTATION = txt;
```

При создании договора с клиентом на основе его анкеты, скопировать в договор личные данные:

```
var dogovor = dest.Versions.Single(x=>x.IsActual);
var anketa = src.Versions.Single(x=>x.IsActual);

dogovor.String_Surname_Client = anketa.strSurname;
dogovor.String_FirstName_Client = anketa.strFirstName;
dogovor.String_SecondName_Client = anketa.strSecondName;
...
```

Скрипты цветовых фильтров

Скрипты, описанные в этой секции определяются только в поисковых запросах и работают на списке-результате запроса.

Доступные переменные:

`dest` – обёртка типа документа, содержащая только те поля, которые были указаны в поисковом запросе, как реквизиты отображения.

`Catalogs` – переменная имеет тип `IScriptHelper`, она нужна для организации взаимодействия скрипта с `eDocLib`'ом.

Скрипт цветовых фильтров, в отличие от остальных должен явно возвращать значения типа `boolean`, т.е. обязательно должен быть явно вызов `return (true или false)` – в зависимости от того, удовлетворяет ли документ условиям, или нет.

В результате выполнения в списке результатов поиска цветом помечаются записи, удовлетворяющие указанным условиям (возвращено значение `true`). Цвет, которым выделяются записи, настраивается в поисковом запросе.

Примеры:

Выделить цветом документы, у которых дата документа попадает в последние 10 дней:

```
var result = dest.Versions.Single(x => x.IsActual).Date >=
DateTime.Today.AddDays(-10);

return result;
```

Выделить цветом документы, автором которых является Иванов:

```
var result = false;

foreach (var row in dest.Versions.Single(x => x.IsActual).Authors)
{
    if (row.Author_Id == Catalogs.GetUolIdByName("Иванов"))
    {
        result = true;
    }
}

return result;
```

Или:

```
return dest.Versions.Single(x => x.IsActual).Authors.Any(a => a.Author_Id ==  
Catalogs.GetUolIdByName("Иванов"));
```

Скрипты на сохранение документа

Скрипты, описанные в этой секции выполняются при сохранении нового или измененного документа. Они позволяют проводить проверки введенных значений, а также дозаполнять сохраняемый документ.

Доступные переменные:

`dest` – переменная имеет тип обёртки типа документа, который указан на группе документов, для которой создаётся скрипт. Т.е. если группа документов создана от типа «Документ» - тип переменной `dest` будет `Eos.DP.Wrapper.Doc.BaseDocument`, если от типа «Поручение», то тип `dest` – `Eos.DP.Wrapper.Doc.Task` и т.д.

`Catalogs` – переменная имеет тип `IScriptHelper`, она нужна для организации взаимодействия скрипта с `eDocLib`'ом.

Пример:

При сохранении нового документа группы «Входящий документ» нужно проставить срок через неделю от текущего момента.

```
var version = dest.Versions.Single(x => x.IsActual);  
  
if (version.IsAdded())  
    version.EndDatePlan = DateTime.Now.AddDays(7);
```

Catalogs

Объект `Catalogs` имеет тип `IScriptHelper`.

Методы:

```
Guid GetDocGroupIdByName(string docGroupName);  
Guid GetUserIdByLogin(string userLogin);  
Guid GetUserIdByName(string userName);  
Guid GetUolIdByName(string uolName);  
Guid GetUserGroupIdByName(string userGroupName);  
Guid GetCurrentUser();  
Guid GetCurrentUserUolId();  
  
Guid GetFolderIdByName(string folderName);  
Guid GetParentFolderIdByName(string folderName);  
  
Guid GetCatalogIdByName(string typeId, string name);  
  
Guid[] GetUserSubstitutes();  
Guid[] GetUserChiefs();  
  
IEnumerable<Guid> GetSelectedIds(string tableName, DataSet buffer);
```

`GetDocGroupByName` – возвращает идентификатор группы документов, аргумент – название группы документов.

`GetUserIdByLogin` и `GetUserIdByName` – возвращают идентификатор пользователя системы, аргументами служат логин пользователя или имя пользователя соответственно. Эти методы используются для работы с секцией ACL (список доступа) документов.

`GetUolIdByName` – возвращает идентификатор контакта (Uol) из справочника контактов.

`GetUserGroupIdByName` – метод, возвращающий идентификатор группы пользователей.

`GetCurrentUserId` и `GetCurrentUserUolId` – возвращают идентификатор пользователя системы и идентификатор контакта текущего пользователя.

`GetFolderIdByName` – возвращает id личной папки, если она доступна пользователю, по имени папки. Папки скрытых пользователей не учитываются.

`GetParentFolderIdByName` – возвращает id личной папки, содержащей указанную по имени, если обе папки доступны пользователю. Папки скрытых пользователей не учитываются.

`GetCatalogIdByName` – возвращает id записи справочника, имеющего указанный тип, по имени записи.

`GetUserSubstitutes` – возвращает заместителей текущего пользователя, - их id из справочника `UolBase`.

`GetUserChiefs` – возвращает начальников текущего пользователя (тех, кого он замещает), - их id из справочника `UolBase`.

`GetSelectedIds` – возвращает массив id выбранных пользователем записей в указанной таблице указанного буфера данных. Рассматривается первый найденный путь к таблице (состоящий из нескольких таблиц), который заканчивается на указанной таблице.

Пример:

Нужно получить идентификатор контакта, например гражданина Иванова И.И.:

```
var ID = Catalogs.GetUolIdByName("Иванов");
```

Примечание: Если метод не нашёл нужный идентификатор, то возвращается `Guid.Empty`.

Для методов, возвращающих массив идентификаторов, пустой результат будет выглядеть как массив с нулевым количеством элементов.

Приложение 5. Печатные формы.

Печатная форма – что это?

Печатная форма представляет собой отчет в формате одного из поддерживаемых провайдеров: Stimulsoft Reports или Reporting Services, источником данных для которого является DataSet ADO.NET с данными документа или результатами поиска. Структура источника данных зависит от типа документа. На этапе создания печатной формы для определенного типа документов описание источника данных в формате XML, специфичное для этого типа, можно получить напрямую из системы, при этом провайдер может использовать это описание как информацию об источнике данных, необходимую для создания отчета.

Что печатаем

Система поддерживает печать карточек документов различных типов – как встроенных, так и пользовательских, а также печать списков, получаемых в результате выполнения поискового запроса. В настоящей версии каждому типу документа или поисковому запросу может быть поставлена в соответствие одна печатная форма. Печатная форма может отображать информацию всех реквизитов документа, в том числе и системных, таких как **Файлы, Рубрики, Ссылки, Обратные ссылки, Список доступа**. Также, возможна печать как нескольких версий документа, так и только текущей. При печати результатов поискового запроса, печатная форма может отображать только те реквизиты, которые выводятся в результатах поиска (закладка «Результаты поиска»).

Как печатаем

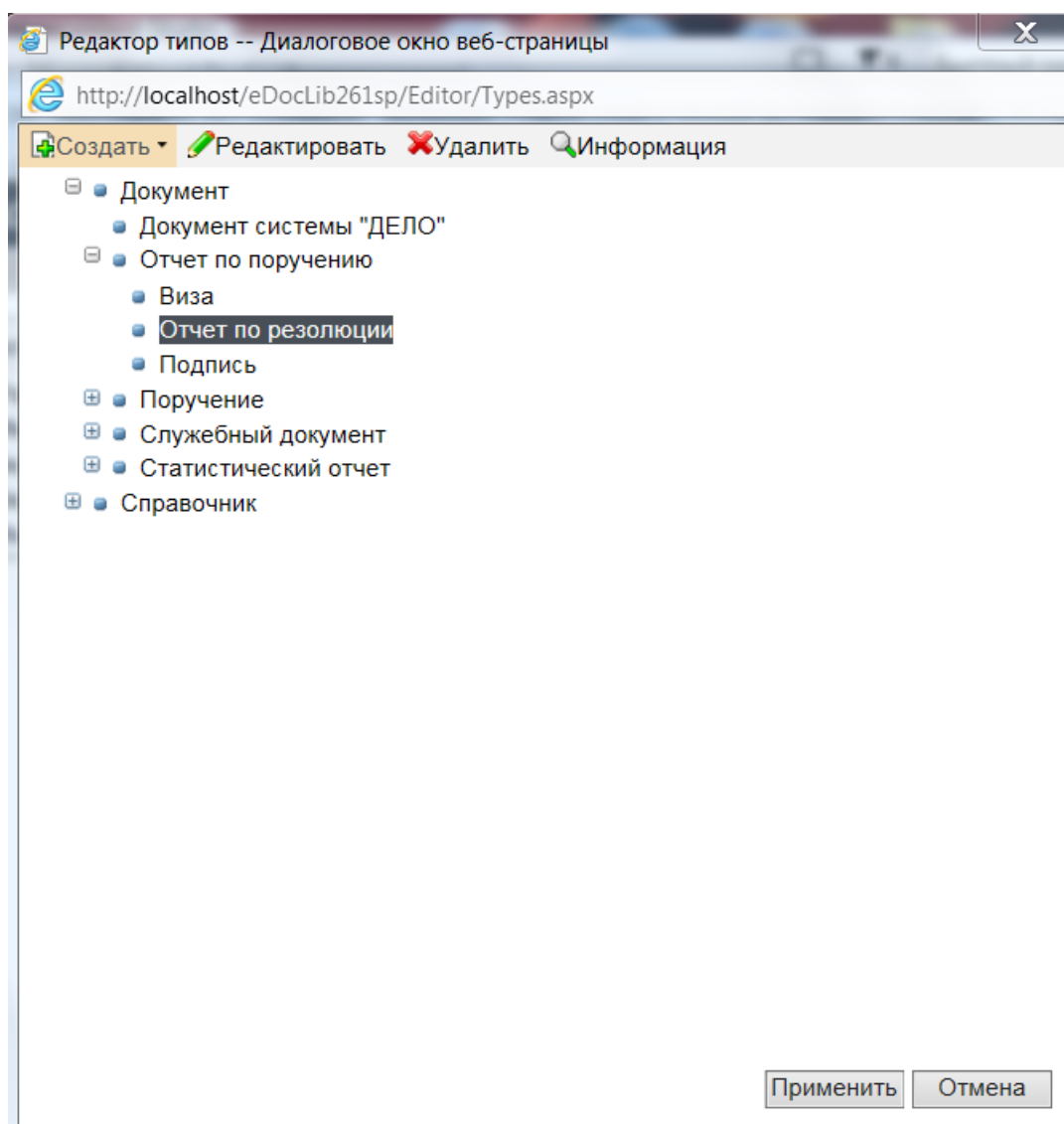
Печать осуществляется из окна просмотра документа, при этом на сервере приложения запускается формирование печатной формы данного типа документов. Результат работы в формате PDF пересылается на клиентский компьютер. Непосредственно печать производится пользователем с помощью Acrobat Reader.

Создание печатной формы для документа

В разделе описан процесс создания печатной формы для документа.

Выбор типа документов

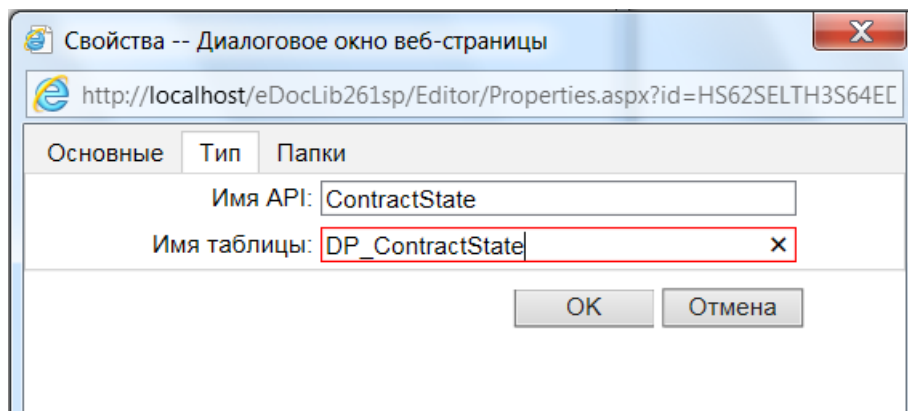
Сперва необходимо определить, для какого типа документов будет создаваться печатная форма. Можно выбрать существующий тип, либо создать новый (см. рис. ниже). Подробнее о Конфигураторе системы «eDocLib», создании новых типов документов и справочников можно прочитать в «Кратком руководстве пользователя».

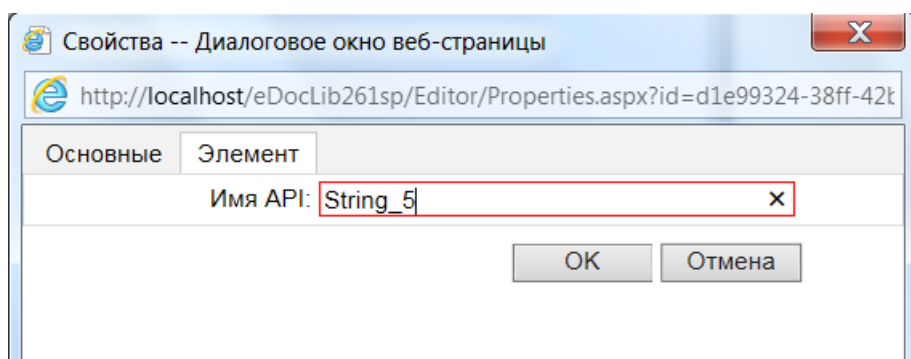


Редактирование API-имен реквизитов в типе документов.

После того, как вы определились с типом документов, следует убедиться, что все пользовательские реквизиты и секции, которые нужно отобразить на печатной форме, содержат понятные вам API-имена. Это важно, т.к. по автоматически создаваемым системой API-именам реквизитов и секций (типа String_5 или String_10) может быть неясно, какие именно реквизиты они обозначают, а в дальнейшем названия колонок в источнике данных отчета будут соответствовать API-именам реквизитов и секций типа документов.

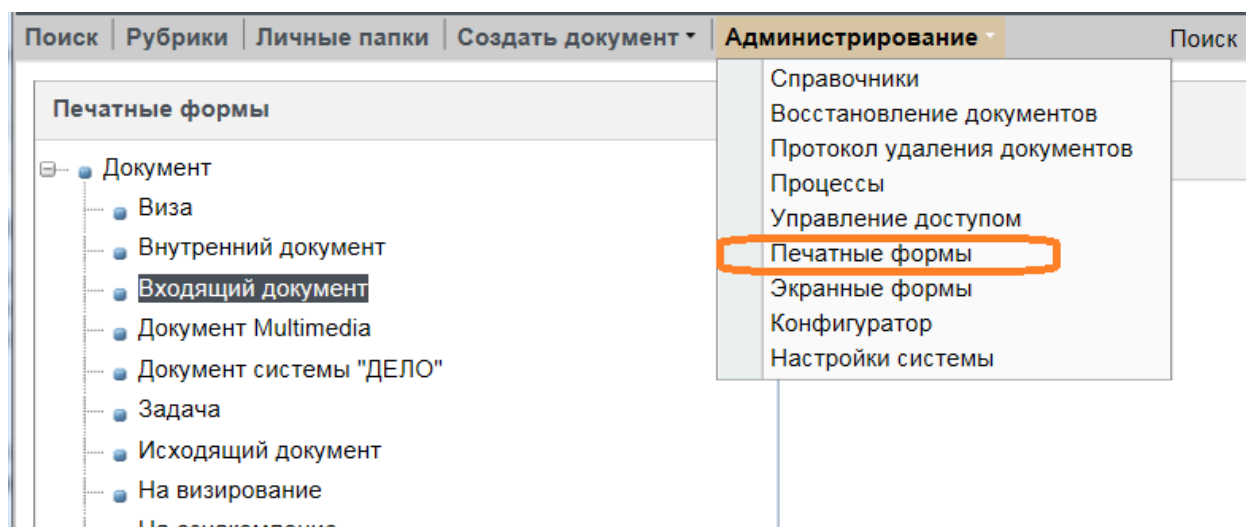
Возможности по изменению API-имен типов документов или реквизитов типов документов имеются в Конфигураторе системы (см. «Краткое руководство пользователя»).



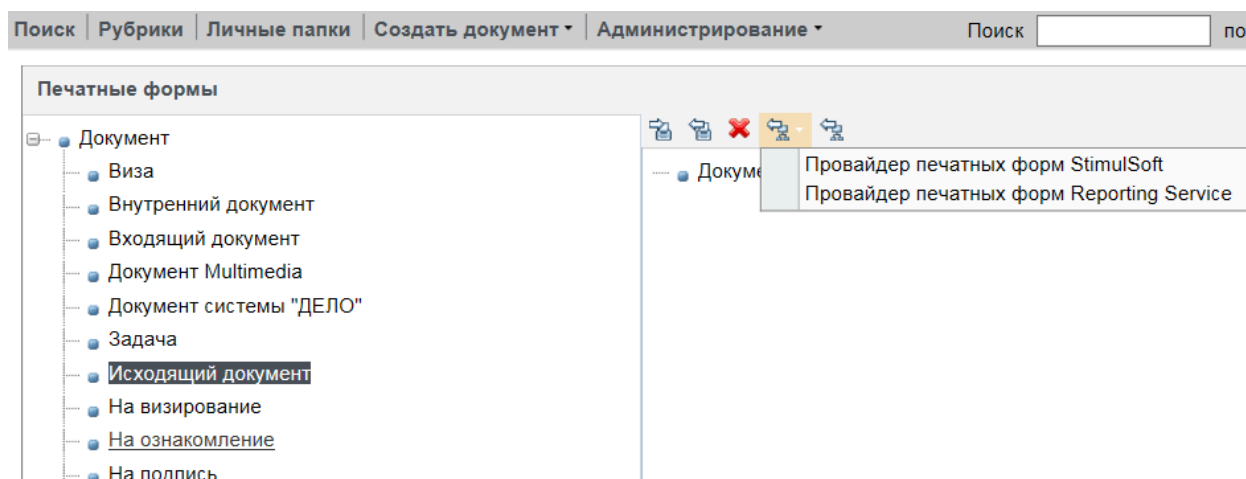


Получение схемы источника данных в XML формате.

Следующий шаг – получение необходимого для провайдера описания источника данных, нужного для создания печатной формы. Необходимый функционал уже встроен в систему и доступен из интерфейса технолога, со страницы Печатных форм.



Далее выбираем нужный тип документа из списка и щелкаем кнопку **Экспортировать схему данных для печатной формы**.



После этого следует выбрать провайдер печатных форм и сохранить файл (schema.xsd или schema.txt).

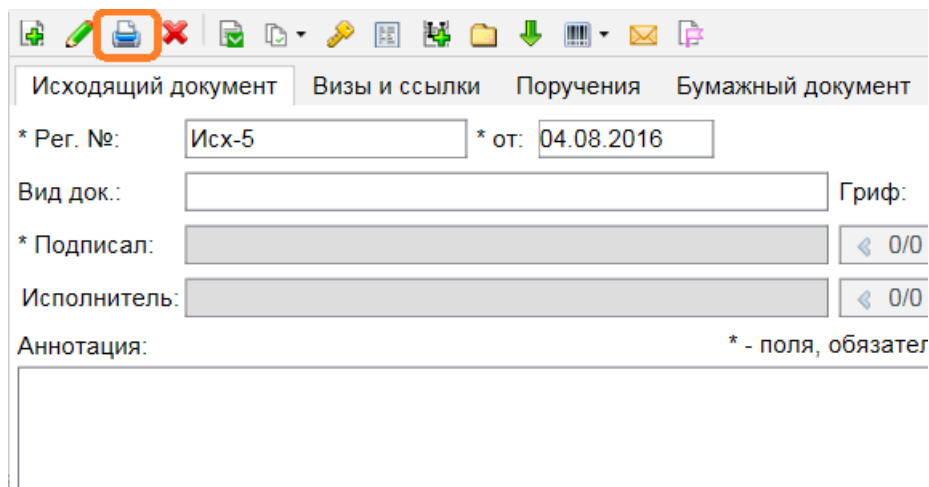
Создание печатной формы

Используя соответствующее программное обеспечение, на основе сохраненной ранее схемы данных необходимо создать печатную форму. Впоследствии она будет импортирована в систему «eDocLib» в качестве печатной формы для выбранного нами типа документов.

Печать документа

Система позволяет выводить на печать документы. Для этого в неё должна быть загружена соответствующая печатная форма. В зависимости от печатной формы, может выводиться как текущая версия, так и все версии.

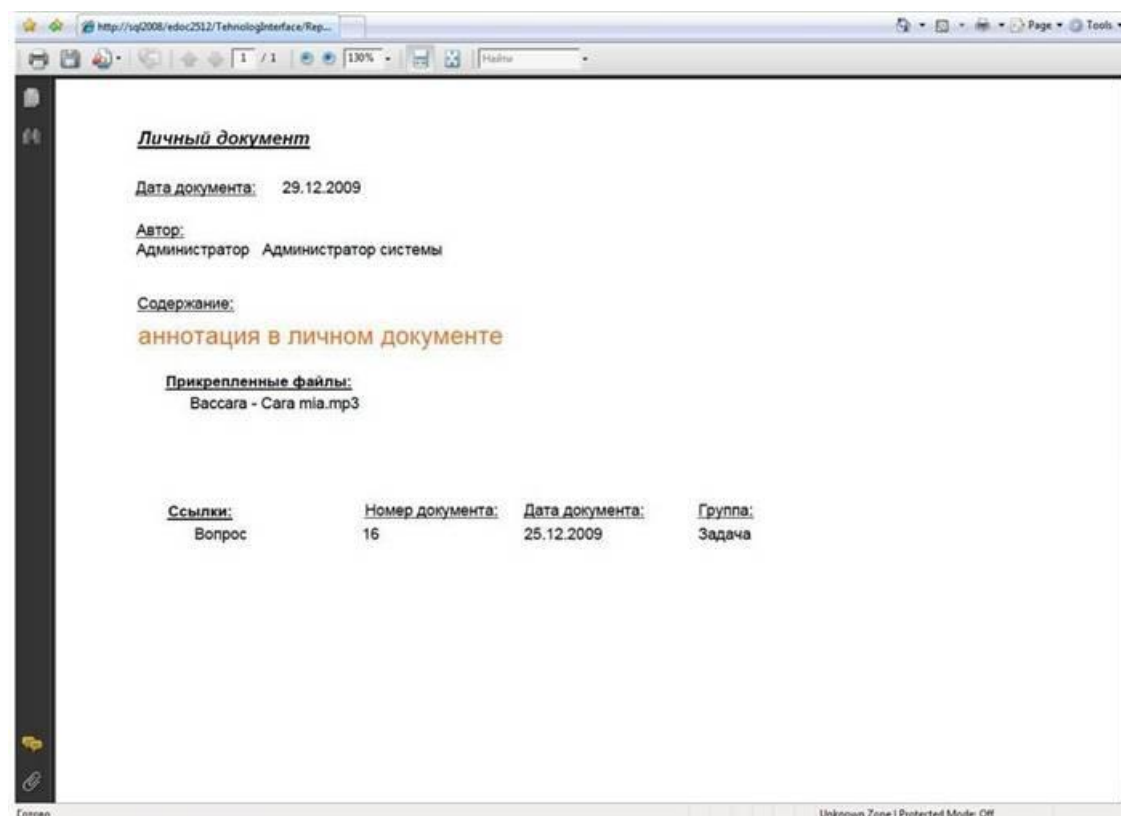
Для печати документа нужно открыть его и нажать кнопку **Печать** (см. рис.).



The screenshot shows a web application interface. At the top is a toolbar with various icons; the print icon (a printer) is highlighted with an orange rectangle. Below the toolbar are four tabs: "Исходящий документ", "Визы и ссылки", "Поручения", and "Бумажный документ". The "Исходящий документ" tab is active. Below the tabs are several input fields and buttons:

- * Рег. №: * от:
- Вид док.:
- Гриф:
- * Подписал:
- Исполнитель:
- Аннотация: * - поля, обязательны

В результате получим готовый к печати документ в формате pdf, например:



The screenshot shows a web browser displaying a PDF document. The address bar shows the URL: <http://vq2008/edoc2512/TehnologInterface/Rep...>. The document content is as follows:

Личный документ

Дата документа: 29.12.2009

Автор:
Администратор Администратор системы

Содержание:
аннотация в личном документе

Прикрепленные файлы:
Baccara - Cara mia.mp3

<u>Ссылки:</u>	<u>Номер документа:</u>	<u>Дата документа:</u>	<u>Группа:</u>
Вопрос	16	25.12.2009	Задача

At the bottom of the browser window, there is a status bar with the text "Unknown Zone | Protected Mode: Off".

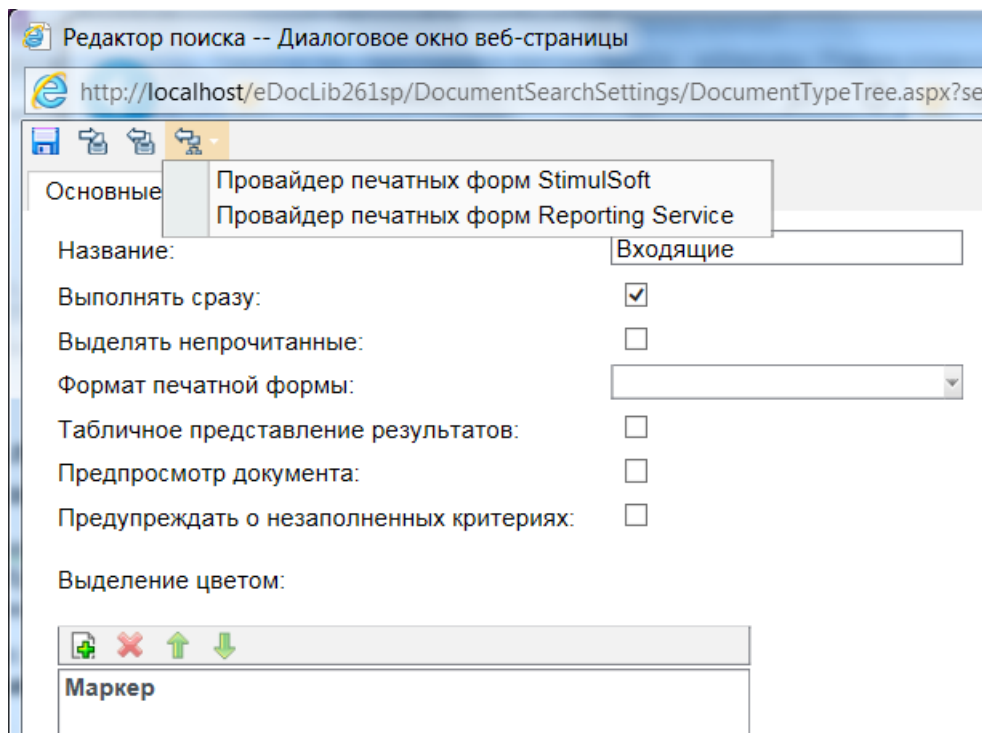
Создание печатной формы поискового запроса

В разделе описан процесс создания печатной формы для поискового запроса.

Получение схемы источника данных в XML формате для поискового запроса

Реализована возможность создавать печатные формы для отображения и печати результатов поиска документов. Для этого требуется получить схему данных для типа документа, по которому производится поиск.

Для этого в редакторе поискового запроса щелкаем кнопку **Экспортировать схему данных для печатной формы** (см. рис.).



После этого следует выбрать провайдер и сохранить файл schema.xsd или schema.txt.

Примечание: Хотя в схеме данных будут присутствовать все реквизиты типа документа, в печатной форме должны быть доступны только те, которые выводятся в результатах поиска (закладка «Результаты поиска»).












Создание печатной формы

Создание печатной формы аналогично описанному выше: используя соответствующее программное обеспечение, на основе сохраненной схемы данных необходимо создать печатную форму, чтобы впоследствии ее импортировать в систему.

Печать результатов поиска

Печать результатов поиска подразумевает печать списка, полученного из поискового запроса. Выводятся только те записи, которые отмечены галочками. Для успешного выполнения в систему должна быть загружена соответствующая печатная форма.

Для печати результатов поиска следует отметить элементы, подлежащие печати, и нажать кнопку **Печать** (см. рис.).

<div>     </div> <div>Номер документа   10 </div>				
<input type="checkbox"/>	№	от	группа	Аннотация
<input checked="" type="checkbox"/> 	1	03.03.2016	Входящий документ	Аннотация
<input checked="" type="checkbox"/> 	10	24.03.2016	Входящий документ	

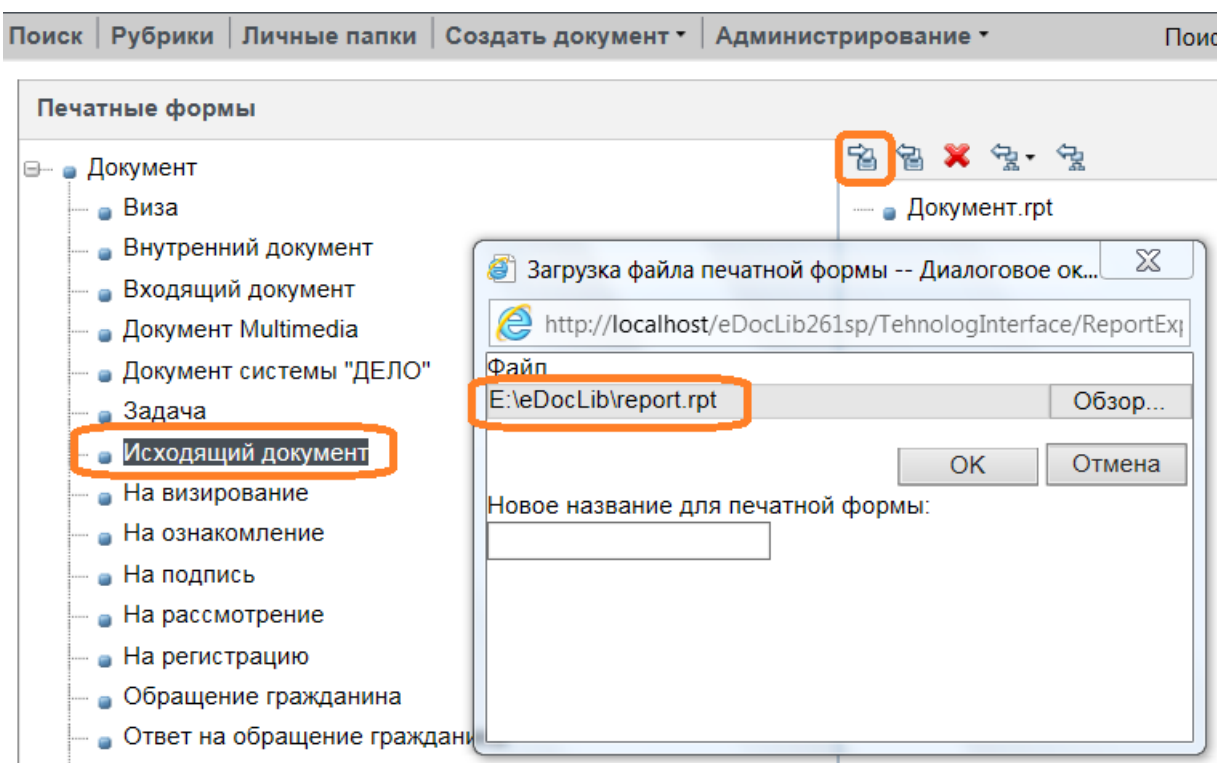
В результате получим готовый к печати документ в формате pdf.

Импорт/экспорт печатных форм

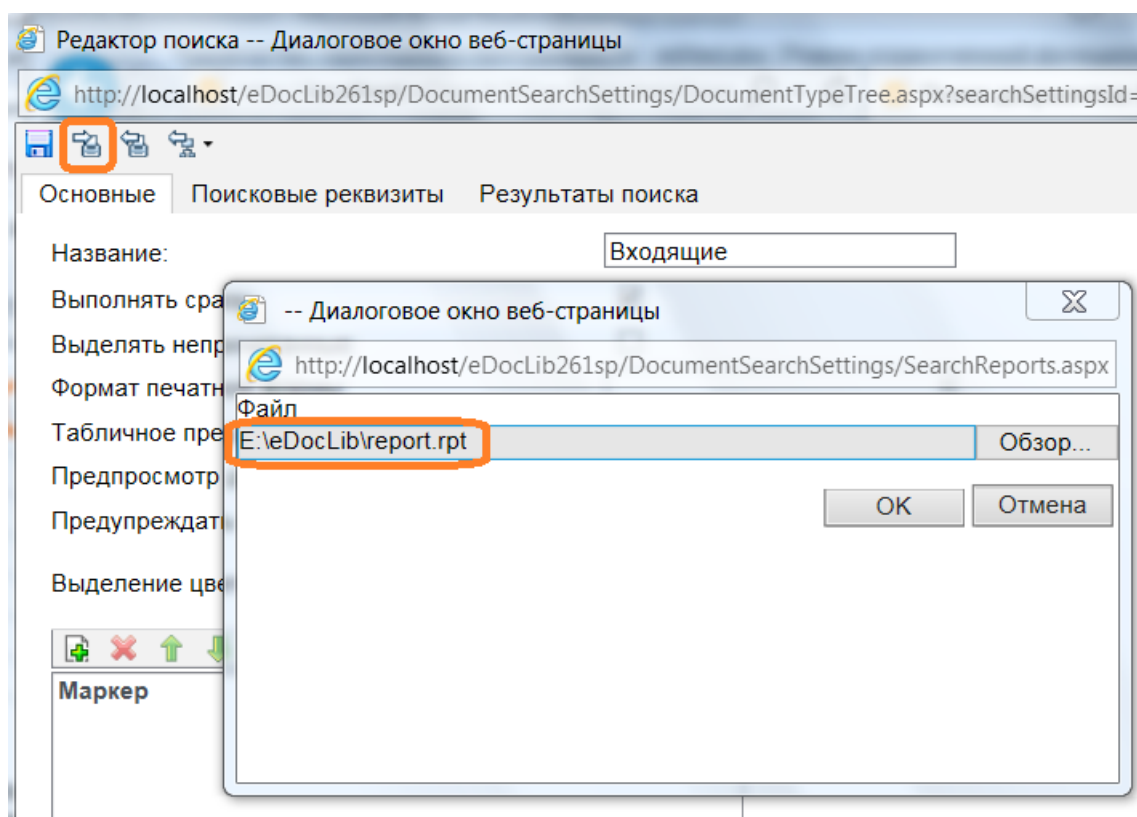
Существует два способа поставить в соответствие типу документа печатную форму. Первый заключается в загрузке печатной формы для этого типа документов через интерфейс технолога. Второй – в импорте конфигурации, уже содержащей печатную форму для этого типа документов. При этом старая печатная форма (если такая присутствует) будет заменена на печатную форму из импортируемой конфигурации. Экспортируемая из системы конфигурация содержит все загруженные в нее печатные формы.

Импорт печатной формы в систему

После того, как отчет StimulSoft или Reporting Service создан, его можно импортировать в систему как печатную форму для того типа документов, для которого мы выгружали схему данных (schema.xsd или schema.txt), используя интерфейс технолога.



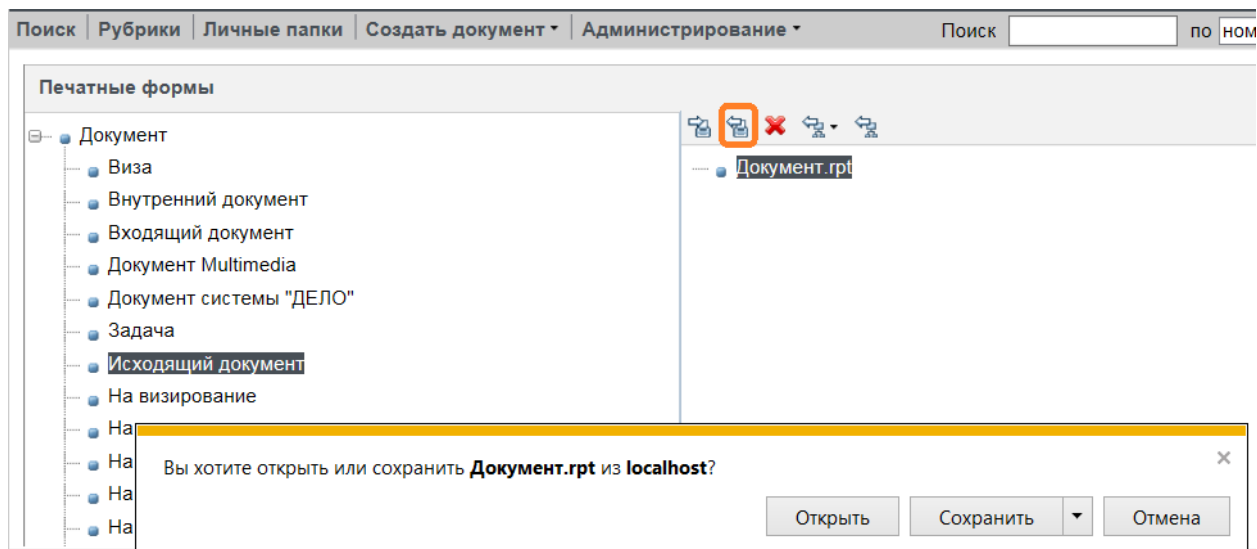
Аналогично загружается печатная форма для поискового запроса из окна редактирования поискового запроса:



Внимание! Печатные формы импортируются вместе с конфигурацией системы. При этом, если для какого-то типа документов печатная форма уже загружена – она будет заменена на импортируемую вместе с конфигурацией.

Экспорт печатной формы из системы

При необходимости внести изменения в печатную форму какого-либо типа документов, ее следует экспортировать из системы, внести необходимые изменения, а затем импортировать заново. Экспорт, как и импорт, осуществляется из интерфейса технолога.



Приложение 6. Триггеры.

Реализована возможность программного расширения бизнес логики системы в плане работы с существующими документами.

Например, если возникает необходимость сделать сложную проверку документа перед сохранением, предзаполнение перед созданием и т.д.

Действия в системе производятся посредством выполнения команд. Например, загрузить документ, сохранить документ, найти документ и т. д. Обычно команда содержит набор событий, которые вызываются в процессе её работы. Набор команд и их синтаксис аналогичен методам API, рассмотренным в данном руководстве.

Например, команда *SaveDocument* содержит 4 события – *BeforeSave*, *BeforeSaveValidate*, *AfterSave*, *AfterSaveValidate*. Соответственно есть возможность подписаться на эти события и выполнить дополнительную работу. Код, который подписывается на события команд и выполняет какую-то работу, называется триггером.

Для написания триггера Вам понадобится **Microsoft Visual Studio 2012** и выше, проект должен быть создан для **.Net 4.5.2**.

Триггер выглядит, как отдельный класс, помеченный атрибутом *RegisterTrigger* и унаследованный от интерфейса *ITrigger*.

```
[RegisterTrigger(typeof(ILoadDocumentCommand))]
public class LoadDocumentCustomContentTrigger : ITrigger
```

В методе *Advise* триггер подписывается на те события, с которыми он будет работать. Например:

```
public void Advise(ICommand command)
{
    //в методе Advise триггер должен подписаться на те события команды,
    //которые его интересуют
    ((ILoadDocumentCommand)command).AfterLoad +=
    LoadDocumentCustomContentTrigger_AfterLoad;
}
```

Регистрация триггера

Для того, чтобы зарегистрировать триггер в системе, нужно скопировать сборку в папку bin сервера приложений. Затем нужно добавить в файл *runtime.config* в секцию *assemblies* информацию о добавляемой и всех зависимых сборках.

Например, регистрируем триггер *BuisnessLogicExtention* (см. раздел Примеры, Пример 1). При создании триггера были использованы сборки *Eos.DP.Core*, *Eos.eDocLib.Business.contracts*, *Eos.DP Wrapper* и *wrappers* – их прописывать не нужно, они уже есть в папке bin и загрузятся сами.

Прописываем сборку в секцию *assemblies*:

```
<assemblies>
  <add assembly="Eos.DP.AppServer" />
  <add assembly="Eos.DP.SearchEngine.Impl" />
  <add assembly="Eos.DP.BusinessLogic.Impl" />
  <add assembly="Eos.eDocLib.Business.Repositories" />
  <add assembly="Eos.DP.PRS.Logic.Impl" />
  <add assembly="Eos.DP.Desktop.Impl" />
  <add assembly="Eos.DP.Core" />
```

```
<add assembly="Eos.DP.ExportImport.Impl" />
<add assembly="Eos.DP.SamlAuthorization.App" />
<add assembly="Eos.eDocLib.Reports.Stimulsoft" />
<add assembly="Eos.eDocLib.Reports.Reporting" />
<add assembly="Eos.eDocLib.DocMobile.Export" />
<add assembly="Eos.eDocLib.DocMobile.BusinessLogic" />
<add assembly="Eos.eDocLib.DocflowExtensions" />
<add assembly="Eos.DP.Management" />
<add assembly="Eos.eDocLib.Web.Controllers" />
<add assembly="BusinessLogicExtension" />
</assemblies>
```

После внесения изменений, сервер приложений нужно перезапустить.

Результат работы триггера:

The screenshot shows the 'Входящий документ' (Incoming Document) form in the Eos application. The form includes fields for 'Пер. №:' (1), '* от:' (03.03.2016), 'Вид документа:', 'Гриф:' (Общий), '* Корреспондент:', 'Исходящий №:', 'от:', 'Подписал:', '* Адресат:', 'Способ получения:', and 'Аннотация:'. The 'Аннотация' field is highlighted with an orange box and contains the text 'АннотацияHello from custom trigger!'.

Какие сборки нужно включить в проект

При разработке триггера, в проект нужно включить как минимум сборки:

Eos.DP.Core,

Eos.eDocLib.Business.Contracts,

Eos.DP Wrapper,

wrappers.

Их достаточно для базовой функциональности триггера. С их помощью можно работать с буфером документа, реализовывать обработчики ошибок (пространство имен *Eos.DP.Exceptions*), вызывать команды.

Как из триггера работать с буфером документа

Работа с буфером документа аналогична работе через API. Смотрите соответствующие разделы настоящего руководства.

Команды, для которых можно создавать триггеры

Данный раздел содержит описание команд, для которых можно создавать триггеры.

ILoadDocumentCommand

Команда загружает документ для просмотра или редактирования.

События:

<code>event System.EventHandler<LoadDocumentCommandEventArgs> AfterLoad</code>	вызывается после загрузки документа
<code>event System.EventHandler<LoadDocumentCommandEventArgs> AfterLoadValidate</code>	вызывается после AfterLoad
<code>event System.EventHandler<LoadDocumentCommandEventArgs> BeforeLoad</code>	вызывается перед загрузкой документа
<code>event System.EventHandler<LoadDocumentCommandEventArgs> BeforeLoadValidate</code>	вызывается после BeforeLoad

События AfterLoadValidate и BeforeLoadValidate следует использовать для реализации проверок значений реквизитов, которые могут быть изменены при обработке событий AfterLoad и BeforeLoad.

Методы:

`System.Data.DataSet`

`Run (Eos.DP.BusinessLogic.Documents.LoadDocumentCommandParameters parameters) –`
выполнение команды.

LoadDocumentCommandEventArgs

Свойства:

`BaseDocument Document`: обёртка документа.

`bool ForUpdate`: флаг, указывающий, был ли документ загружен для изменения или нет. Если флаг установлен, то документ считается заблокированным, и другие пользователи уже не смогут взять его на редактирование пока текущий пользователь его не сохранит.

`EnumOfGuid CallerType`: один или несколько атрибутов, взятых из `CommandCallerType`, отражающих источник (контекст) вызова команды / триггера. Класс `CommandCallerType` находится в сборке `Eos.DP.Core`. Пример: `if (e.CallerType.HasFlag(CommandCallerType.API)) {...}`

`bool IsSafe`: флаг, указывающий, идет ли вызов команды (на которую подписан триггер) из другой команды. Положительное значение говорит в том числе о том, что все проверки на права уже были произведены.

`DataSet Buffer`: буфер документа, над которым построена обертка.

ISaveDocumentCommand

Команда сохраняет заданный документ.

События:

<code>event System.EventHandler<SaveDocumentCommandEventArgs> AfterSave</code>	вызывается после сохранения документа
<code>event System.EventHandler<SaveDocumentCommandEventArgs> AfterSaveValidate</code>	вызывается после AfterSave
<code>event System.EventHandler<SaveDocumentCommandEventArgs> BeforeSave</code>	вызывается перед сохранением документа
<code>event System.EventHandler<SaveDocumentCommandEventArgs> BeforeSaveValidate</code>	вызывается после BeforeSave

События AfterSaveValidate и BeforeSaveValidate следует использовать для реализации проверок значений реквизитов, которые могут быть изменены при обработке событий AfterSave и BeforeSave.

Методы:

`System.Data.DataSet`

`Run(Eos.DP.BusinessLogic.Documents.SaveDocumentCommandParameters parameters)`

– выполнение команды.

SaveDocumentCommandEventArgs

Свойства:

<code>FastACLClass</code> ACL : права на документ текущего пользователя (сохраняющего документ)
<code>BaseDocument</code> Document : обёртка типа документа
<code>DocumentGroup</code> Group : группа документа
<code>SaveDocumentCommandParameters</code> Parameters : параметры сохранения документа, содержащие единственное свойство – буфер документа
<code>bool</code> IsSafe : флаг, указывающий, идет ли вызов команды (на которую подписан триггер) из другой команды. Положительное значение говорит в том числе о том, что все проверки на права уже были произведены.
<code>EnumOfGuid</code> CallerType : один или несколько атрибутов, взятых из <code>CommandCallerType</code> , отражающих источник (контекст) вызова команды / триггера. Класс <code>CommandCallerType</code> находится в сборке Eos.DP.Core. Пример: <code>if (e.CallerType.HasFlag(CommandCallerType.API)) {...}</code>
<code>ActionType</code> Action : флаг, говорящий о том, что происходит с документом: добавление, обновление, удаление или ничего.
<code>ActionInfoCollection</code> Updates : список действий, произведенных над документом, каждое из которых содержит старое (OldValue) и новое (NewValue) значение, путь к нему (PathString) и тип действия (Action).

ICreateDocumentByGroupCommand

Команда создает документ заданной группы.

События:

<code>event System.EventHandler<CreateDocumentCommandEventArgs> AfterCreate</code>	вызывается после создания документа
<code>event System.EventHandler<CreateDocumentCommandEventArgs> AfterCreateValidate</code>	вызывается после AfterCreate
<code>event System.EventHandler<CreateDocumentCommandEventArgs> BeforeCreate</code>	вызывается перед созданием документа
<code>event System.EventHandler<CreateDocumentCommandEventArgs> BeforeCreateValidate</code>	вызывается после BeforeCreate

События AfterCreateValidate и BeforeCreateValidate следует использовать для реализации проверок значений реквизитов, которые могут быть изменены при обработке событий AfterCreate и BeforeCreate.

Методы:

`Eos.DP.Wrapper.Doc.BaseDocument Run(System.Guid groupId)` — выполнение команды.

CreateDocumentCommandEventArgs**Свойства:**

Guid ExtendedCreationKey: если задан, ссылки на рутовый документ не будет создано. Если задан, по значению будет получен через LoadSessionData буфер рутового документа.

BaseDocument Document: создаваемый документ

DocumentGroup Group: группа создаваемого документа

Guid GroupId: идентификатор группы создаваемого документа

BaseDocument Root: документ, на который проставляется ссылка.

EnumOfGuid CallerType: один или несколько атрибутов, взятых из `CommandCallerType`, отражающих источник (контекст) вызова команды / триггера.

bool IsSafe: флаг, указывающий, идет ли вызов команды (на которую подписан триггер) из другой команды.

Guid ScriptId: id скрипта копирования для связанного документа, получаемый из группы Root документа (на который проставляется ссылка), применяемый на создаваемом документе

Dictionary<string, object> Other: если ключ равен PWR, а значение - Guid (в команду соответствующим по счету параметром может быть передано и просто значение, тогда ключ будет создан PWR), и если создается поручение, то id родительского поручения выставляется этому Guid в триггере на AfterCreate. Используется только для этого одного случая и ключа. Раньше использовалось 2 параметра по 2м разным ключам.

Команды, которые можно вызывать из триггера

В триггере можно не только обрабатывать события определенных команд, но и запускать другие команды.

В аргументах события, на которые подписывается триггер, есть свойство `ICommandExecutionContext` `Context`, которое используется в том числе для вызова других команд через методы `ExecuteCommand`.

Пример:

```
DataSet result = e.Context.ExecuteCommand<DataSet>(
    name: "SearchCatalog",
    safe: true,
    p: new object[] { DocumentGroup.TypeId, "_Id:=(Name,?)", "Документ" });
```

Некоторые важные команды, которые можно вызывать из триггера:

SearchCatalog, LoadCatalog, SaveCatalog, LoadDocument, SaveDocument, CreateDocumentByGroup.

SearchCatalog

Команда выполняет поиск записей справочника.

События:

<code>event System.EventHandler<SearchCatalogCommandEventArgs></code> <code>BeforeSearchParamsProcessing</code>	вызывается перед обработкой поискового критерия
--	---

Методы:

`System.Data.DataSet`
`Run(Eos.DP.BusinessLogic.Catalogs.SearchCatalogCommandParameters parameters)`
– выполнение команды.

SearchCatalogCommandParameters

Параметры команды поиска в справочнике.

```
public SearchCatalogCommandParameters(string typeId, string query, params
object[] values)
```

Параметры:

<code>typeId:</code>	тип справочника
<code>query:</code>	строка поискового запроса
<code>values:</code>	список параметров переменной длины

Подробнее см. приложение 2.

LoadCatalog

Команда загружает для чтения экземпляр справочника заданного типа.

События:

<code>event System.EventHandler<LoadCatalogCommandEventArgs></code> <code>AfterLoad</code>	вызывается после загрузки экземпляра справочника
<code>event System.EventHandler<LoadCatalogCommandEventArgs></code> <code>AfterLoadValidate</code>	вызывается после <code>AfterLoad</code>
<code>event System.EventHandler<LoadCatalogCommandEventArgs></code> <code>BeforeLoad</code>	вызывается до загрузки экземпляра справочника
<code>event System.EventHandler<LoadCatalogCommandEventArgs></code>	

BeforeLoadValidate

События AfterLoadValidate и BeforeLoadValidate следует использовать для реализации проверок значений реквизитов, которые могут быть изменены при обработке событий AfterLoad и BeforeLoad.

Методы:

System.Data.DataSet

Run(Eos.DP.BusinessLogic.Catalogs.LoadCatalogCommandParameters parameters) – выполнение команды.

LoadCatalogCommandParameters

Параметры команды сохранения документа.

<code>public LoadCatalogCommandParameters(System.Guid id, string typeId)</code>

Параметры:

id:	идентификатор экземпляра
typeId:	строковый идентификатор типа справочника

SaveCatalog

Команда сохраняет, добавляет или удаляет запись справочника.

События:

<code>event System.EventHandler<SaveCatalogCommandEventArgs> AfterSave</code>	вызывается после сохранения записи
<code>event System.EventHandler<SaveCatalogCommandEventArgs> AfterSaveValidate</code>	вызывается после AfterSave
<code>event System.EventHandler<SaveCatalogCommandEventArgs> BeforeSave</code>	вызывается перед сохранением записи
<code>event System.EventHandler<SaveCatalogCommandEventArgs> BeforeSaveValidate</code>	вызывается после BeforeSave

События AfterSaveValidate и BeforeSaveValidate следует использовать для реализации проверок значений, которые могут быть изменены при обработке событий AfterSave и BeforeSave.

Методы:

System.Data.DataSet

Run(Eos.DP.BusinessLogic.Catalogs.SaveCatalogCommandParameters parameters) – выполнение команды.

SaveCatalogCommandParameters

Параметры команды сохранения записи справочника.

<code>public SaveCatalogCommandParameters(Eos.DP Wrapper.Catalog wrapper)</code>
<code>public SaveCatalogCommandParameters(System.Data.DataSet buffer)</code>

Параметры:

wrapper:	обертка типа справочника
buffer:	DataSet, содержащий изменения

Примеры

В этом разделе приведены 2 демонстрационных триггера.

Пример 1. При открытии документа для просмотра, триггер добавляет в аннотацию строку «Hello from custom trigger!».

```
using System.Linq;
using Eos.DP.Core;
using Eos.DP.BusinessLogic.Documents;

namespace BusinessLogicExtension
{
    // Триггер - это такой класс, код которого будет вызван по совершению
    // определенного события в команде
    // Триггер обязательно должен быть помечен атрибутом RegisterTrigger. В
    // качестве параметра атрибута выступает тип интерфейса команды
    // В данном примере мы будем писать триггер на команду загрузки
    // документа
    [RegisterTrigger(typeof(ILoadDocumentCommand))]
    public class LoadDocumentCustomContentTrigger : ITrigger // триггер
    // должен быть обязательно унаследован от интерфейса ITrigger
    {
        #region ITrigger Members

        public void Advise(ICommand command)
        {
            //в методе Advise триггер должен подписаться на те события
            //команды, которые его интересуют
            ((ILoadDocumentCommand) command).AfterLoad +=
            LoadDocumentCustomContentTrigger_AfterLoad;
        }

        void LoadDocumentCustomContentTrigger_AfterLoad(object sender,
        LoadDocumentCommandEventArgs e)
        {
            // реализуем следующую логику - если документ загружается
            // для просмотра -
            // добавим в аннотацию актуальной версии строку приветствия
            if (!e.ForUpdate)
                e.Document.Versions.Single(x => x.IsActual).ANNOTATION
                += "Hello from custom trigger!";
        }

        #endregion
    }
}
```

Результат работы триггера:

Поиск | Рубрики | Личные папки | Создать документ ▾ | Администрирование ▾

Входящий документ | Бумажный документ | Обсуждение | Ссылки | Файлы

Пер. №: * от:

Вид документа: Гриф:

* Корреспондент: << 0/0 >>

Исходящий №: от:

Подписал:

* Адресат: << 0/0 >>

Способ получения:

Аннотация:

АннотацияHello from custom trigger!

Пример 2. Этот пример демонстрирует более широкие возможности триггера, такие как вызов других команд и обработка ошибок. После сохранения документа, триггер будет создавать связанный документ на основе текущего.

```
using System;
using System.Linq;
using Eos.DP.Core;
using Eos.DP.BusinessLogic.Documents;
using System.Data;
using Eos.DP.BusinessLogic.Catalogs;
using Eos.DP Wrapper;
using Eos.DP.Exceptions;
using Eos.DP Wrapper.Doc;

namespace BusinessLogicAdvancedExtension
{
    //Этот проект отличается от предыдущего в основном тем, что триггер в
    //предыдущем проекте не обращался к другим командам eDocLib, а
    //работал только на загружаемом буфере документа. Этот же триггер во всю
    //использует команды аппсервера.
    [RegisterTrigger(typeof(ISaveDocumentCommand))]
    public class SaveDocumentCreateLinkedTrigger : ITrigger
    {
        #region ITrigger Members

        public void Advise(ICommand command)
        {
            ((ISaveDocumentCommand)command).AfterSave +=
            SaveDocumentCreateLinkedTrigger_AfterSave;
        }

        private static readonly string DOC_GROUP_NAME = "Документ";

        void SaveDocumentCreateLinkedTrigger_AfterSave(object sender,
        SaveDocumentCommandEventArgs e)
        {
            //в этом примере мы будем создавать новый связанный документ
            //после сохранения существующего
        }
    }
}
```

```

        //получим идентификатор группы документов "Документ". Он нам
        //пригодится для создания документа этой группы.
        DataSet searchResult =
e.Context.ExecuteCommand<DataSet>("SearchCatalog",
        true,
        new SearchCatalogCommandParameters(DocumentGroup.TypeId,
            "_Id:=(Name,?)" /* синтаксис см. в Приложении 2 */,
            DOC_GROUP_NAME));

        if (searchResult.Tables[0].Rows.Count == 0)
        {
            //мы не нашли группы документов с таким именем
            //возможные варианты действия - ничего не делать
            //либо - выдать сообщение об ошибке
            //класс сообщения об ошибке обязательно должен быть
            //BusinessLogicExtensionException или являться его потомком
            throw new BusinessLogicExtensionException("В системе
            отсутствует группа документов '{0}'", DOC_GROUP_NAME);
        }

        Guid docGroupId =
searchResult.Tables[0].Rows[0].Field<Guid>("_Id");

        //избегаем рекурсии. Новые документы будут порождаться при
        //сохранении документов с группой отличной от создаваемой
        if (e.Group._Id == docGroupId)
            return;

        //проверим, что мы имеем право создавать документы этой группы
        //получим список доступных групп документов для регистрации
        DataSet docGroups =

        e.Context.ExecuteCommand<DataSet>("GetRegistrationDocumentGroups", true);

        if (docGroups.Tables[0].AsEnumerable().Select(row =>
row.Field<Guid>("DocGroup_Id")).Where(id => id == docGroupId).Count() == 0)
        {
            //если в списке доступных для регистрации групп нет
            //группы, которую мы хотим регистрировать - кинем ошибку
            throw new BusinessLogicExtensionException("Вы не можете
            регистрировать документы группы '{0}'", DOC_GROUP_NAME);
        }

        //создадим новый документ
        BaseDocument newDoc = new BaseDocument(

        e.Context.ExecuteCommand<DataSet>("CreateDocumentByGroup", true,
docGroupId));

        BaseDocument_Version actualVersion = newDoc.Versions.Single(x
=> x.IsActual);

        //добавим в созданный документ ссылку на сохраняемый
        BaseDocument_Version_Link newLink =
actualVersion.AddLinksRow();
        newLink.Doc_Id = e.Document._Id;

        newLink.LinkType_Id = LinkType.Question;
        // получаем и прописываем имя ссылки
        DataSet linkResult =
e.Context.ExecuteCommand<DataSet>("SearchCatalog",
        true,
        new SearchCatalogCommandParameters(LinkType.TypeId,
            "LinkName:=( Id,?)" ,
            newLink.LinkType_Id));
        newLink.LinkName = (string)linkResult.Tables[0].Rows[0][0];

```



```
        /// прописываем группу документа, на который ссылаемся
        newLink.DocGroup_Id = docGroupId;
        newLink.DocGroup_Name = DOC_GROUP_NAME;

        ///скопируем аннотацию с сохраняемого документа в новый
        actualVersion.ANNOTATION = e.Document.Versions.Single(x =>
x.IsActual).ANNOTATION;

        actualVersion.Number = "На основе " +
e.Document.Versions.Single(x => x.IsActual).Number;
        ///сохраним документ
        e.Context.ExecuteCommand("SaveDocument", newDoc.Buffer);
    }

    #endregion
}
}
```